《数学机械化丛书》获国家基础研究发展规划项目“数学机械化方法及其在信息技术中的应用”与“数学机械化应用推广专用经费”资助

数学机械化丛书　7

方程求解与机器证明

——基于MMP的问题求解

高小山　王定康　裘宗燕　杨　宏　著

科 学 出 版 社

北　京

内 容 简 介

本书首先在前三章介绍了数学机械化软件平台 MMP 的基本功能与使用方法，然后在后面的各章中通过 MMP 的运行实例介绍了数学机械化的基本理论与最新近展，特别是方程求解与机器证明方面的最新研究成果.第四章介绍了多项式方程系统,常微分方程系统,偏微分方程系统的吴特征列方法与投影定理.第五章介绍初等与微分几何中定理自动证明与自动发现的吴方法与若干最新进展.第六章介绍代数方程求解的吴特征列方法以及参数方程求解、预解式理论及其在机器人、曲面拼接、代数簇隐式化中的应用.第七章介绍微分方程求解的吴特征列方法以及微分方程初等函数解、行波解、幂级数解的求解方法.第八章介绍代数系统全局优化的吴有限核定理以及不等式的自动证明与发现.每章末尾还对本章的内容与 MMP 实现的方法所涉及的文献进行了介绍.

本书既可以作为 MMP 的使用手册，又可作为科研人员、教师与研究生了解数学机械化基本方法与最新成果的参考书.

图书在版编目(CIP)数据

方程求解与机器证明——基于 MMP 的问题求解/高小山等著. —北京：科学出版社,2006.9

(数学机械化丛书/吴文俊主编)

ISBN 7-03-017862-9

Ⅰ.方…　Ⅱ.高…　Ⅲ.方程解　Ⅳ.O122.2

中国版本图书馆 CIP 数据核字(2006) 第 096715 号

责任编辑：吕　虹　赵彦超／责任校对：纪振红
责任印制：安春生／封面设计：王　浩

科学出版社 出版
北京东黄城根北街 16 号
邮政编码：100717
http://www.sciencep.com

中国科学院印刷厂印刷
科学出版社发行　各地新华书店经销
*
2006 年 9 月第　一　版　开本：B5(720×1000)
2006 年 9 月第一次印刷　印张：18 1/2
印数：1—2 500　字数：342 000

定价：55.00 元
(如有印装质量问题，我社负责调换〈科印〉)

《数学机械化丛书》前言 ①

十六七世纪以来, 人类历史上经历了一场史无前例的技术革命, 出现了各种类型的机器, 取代各种形式的体力劳动, 使人类进入一个新时代. 几百年后的今天, 电子计算机已可开始有条件地代替一部分特定的脑力劳动, 因而人类已面临另一场更宏伟的技术革命, 处在又一个新时代的前夕. 数学是一种典型的脑力劳动, 它在这一场新的技术革命中, 无疑将扮演一个重要的角色. 为了了解数学在当前这场革命中所扮演的角色, 就应对机器的作用, 以及作为数学的脑力劳动的方式, 进行一定的分析.

1. 什么是数学的机械化

不论是机器代替体力劳动, 或是计算机代替某种脑力劳动, 其所以成为可能, 关键在于所需代替的劳动已经"机械化", 也就是说已实现了刻板化或规格化. 正因为割麦、刈草、纺纱、织布的动作已经是机械化刻板化了的, 因而可据此造出割麦机、刈草机、纺纱机、织布机来. 也正因为加减乘除开方等运算这一类脑力劳动, 几千年来就已经是机械地刻板地进行的, 才有可能使得 17 世纪的法国数学家 Pascal, 利用齿轮传动造出了第一台机械计算机 —— 加法机, 并由 Leibniz 改进成为也能进行乘法的机器. 数学问题的机械化, 就要求在运算或证明过程中, 每前进一步之后, 都有一个确定的、必须选择的下一步, 这样沿着一条有规律的、刻板的道路, 一直达到结论.

在中小学数学的范围里, 就有着不少已经机械化了的课题. 除了四则、开方等运算外, 解线性联立方程组就是一个很好的例子. 在中学用的数学课本中, 往往介绍解线性方程组的各种"消去法", 其求解过程是一个按一定程序进行的计算过程, 也就是一种机械的、刻板的过程. 根据这一过程编成程序, 由电子计算机付诸实施, 就可以不仅机器化而且达到自动化, 在几分钟甚至几秒钟之内求出一个未知数多至上百个的线性方程组的解答来, 这在手工计算几乎是不可能的. 如果用手工计算, 即使

---

① 20 世纪七八十年代之交, 我尝试用计算机证明几何定理取得成功, 由此提出了数学机械化的设想. 先后在一些通俗报告与写作中, 解释数学机械化的意义与前景, 例如 1978 年发表于《自然辩证法通讯》的"数学机械化问题"以及 1980 年发表于《百科知识》的"数学的机械化". 二文都重载于 1995 年由山东教育出版社出版的《吴文俊论数学机械化》一书. 经过 20 多年众多学者的努力, 数学机械化在各个方面都取得了丰富多彩的成就, 并已出版了多种专著, 汇集成现在的数学机械化丛书. 现据 1980 年的《百科知识》的"数学的机械化"一文, 稍加修改并作增补, 以代丛书前言.

是解只有三四个未知数的方程组, 也将是繁琐而令人厌烦的. 现代化的国防、经济建设中, 大量出现的例如网络一类的问题, 往往可归结为求解很多未知数的线性方程组. 这使得已经机械化了的线性方程解法在四个现代化中起着一种重要作用.

即使是不专门研究数学的人们, 也大都知道, 数学的脑力劳动有两种主要形式：数值计算与定理证明 (或许还应包括公式推导, 但这终究是次要的). 著名的数理逻辑学家美国洛克菲勒大学教授王浩先生在一篇有名的《向机械化数学前进》的文章中, 曾列举了这两种数学脑力劳动的若干不同之点. 我们可以简略而概括地把它们对比一下：

计算	证明
易	难
繁	简
刻板	灵活
枯燥	美妙

计算, 如已经提到过的加、减、乘、除、开方与解线性方程组, 其所以虽繁而易, 根本原因正在于它已经机械化. 而证明的巧而难, 是大家都深有体会的, 其根本原因也正在于它并没有机械化. 例如, 我们在中学初等几何定理的证明中, 就经常要依靠诸如直观、洞察、经验以及其他一些模糊不清的原则, 去寻找捷径.

2. 从证明的机械化到机器证明

一个值得提出的问题是：定理的证明是不是也能像计算那样机械化, 因而把巧而难的证明, 化为计算那样虽繁而易的劳动呢? 事实上, 这一证明机械化的设想, 并不始自今日, 它早就为 17 世纪时的大哲学家、大思想家和大数学家 Descartes 和 Leibniz 所具有. 只是直到 19 世纪末, Hilbert(德国数学家, 1862～1943) 等创立并发展了数理逻辑以来, 这一设想才有了明确的数学形式. 又由于 20 世纪 40 年代电子计算机的出现, 才使这一设想的实现有了现实可能性.

从 20 世纪二三十年代以来, 数理逻辑学家们对于定理证明机械化的可能性进行了大量的理论探讨, 他们的结果大都是否定的. 例如 Gödel 等的一条著名定理就说, 即使看来最简单的初等数论这一范围, 它的定理证明的机械化也是不可能的. 另一方面, 1950 年波兰数学家 Tarski 则证明了初等几何 (以及初等代数) 这一范围的定理证明, 却是可以机械化的. 只是 Tarski 的结果近于例外, 在初等几何及初等代数以外的大量结果都是反面的, 即机械化是不可能的. 1956 年以来美国开始了利用电子计算机做证明定理的尝试. 1959 年王浩先生设计了一个机械化方法, 用计算机证明了 Russell 等著的《数学原理》这一经典著作中的几百条定理, 只用

了 9 分钟, 在数学与数理逻辑学界引起了轰动. 一时间, 机器证明的前景似乎非常乐观. 例如 1958 年时就有人曾经预测：在 10 年之内计算机将发现并证明一个重要的数学新定理. 还有人认为, 如果这样, 则不仅许多著名哲学家与数学家如 Peano、Whitehead、Russell、Hilbert 以及 Turing 等人的梦想得以实现, 而且计算将成为科学的皇后, 人类的主人!

然而, 事情的发展却并不如预期那样美好. 尽管在 1976 年, 美国的 Hanker 等人, 在高速计算机上用了 1200 小时的计算时间, 解决了数学家们 100 多年来所未能解决的一个著名难题 —— 四色问题, 因此而轰动一时, 但是, 这只能说明计算机作为定理证明的辅助工具有着巨大潜力, 还不能认为这样的证明就是一种真正的机器证明. 用王浩先生的说法, Hanker 等关于四色定理的证明是一种使用计算机的特例机证, 它只适用于四色这一特殊的定理, 这与所谓基础机器证明之能适用于一类定理者有别. 后者才真正体现了机械化定理证明, 进而实现机器证明的实质. 另一方面, 在真正的机械化证明方面, 虽然 Tarski 在理论上早已证明了初等几何的定理证明是能机械化的, 还提出了据以造判定机也即是证明机的设想, 但实际上他的机械化方法非常繁, 繁到不可收拾, 因而远远不是切实可行的.1976 年时, 美国做了许多在计算机上证明定理的实验, 在 Tarski 的初等几何范围内, 用计算机所能证明的只是一些近于同义反复的“儿戏式”的“定理”. 因此, 有些专家曾经发出过这样悲观的论调: 如果专依靠机器, 则再过 100 年也未必能证明出多少有意义的新定理来.

3. 一条切实可行的道路

1976 年冬, 我们开始了定理证明机械化的研究.1977 年春取得了初步成果, 证明初等几何主要一类定理的证明可以机械化. 在理论上说来, 我们的结果已包括在 Tarski 的定理之中. 但与 Tarski 的结果不同, 我们的机械化方法是切实可行的, 即使用手算, 依据机械化的方法逐步进行, 虽然繁复, 也可以证明一些艰深的定理.

我们的方法主要分两步, 第一步是引进坐标, 然后把需证定理中的假设与终结部分都用坐标间的代数关系来表示. 我们所考虑的定理局限于这些代数关系都是多项式等式关系的范围, 例如平行、垂直、相交、距离等关系都是如此. 这一步可以叫做几何的代数化. 第二步是通过代表假设的多项式关系把终结多项式中的坐标逐个消去, 如果消去的结果为零, 即表明定理正确, 否则再作进一步检查. 这一步完全是代数的, 即用多项式的消元法来验证.

上述两步都可以机械与刻板地进行. 根据我们的机械化方法编成程序, 以在计算机上实现机器证明, 并无实质上的困难. 事实上数学所某些同志以及国外的王浩先生都曾在计算机上试行过. 我们自己也曾在国产的长城 203 台式机上证明了像 Simson 线那样不算简单的定理.1978 年初我们又证明了初等微分几何中主要的一

类定理证明也可以机械化. 而且这种机械化方法也是切实可行的, 并据此用手算证明了不算简单的一些定理.

从我们的工作中可以看出, 定理的机械化证明, 往往极度繁复, 与通常既简且妙的证明形成对照, 这种以量的复杂来换取质的困难, 正是利用计算机所需要的.

在电子计算机如此发展的今天, 把我们的机械化方法在计算机上实现不仅不难, 而且有一台微型的台式机也就够了. 就像我们曾经使用过的长城 203, 它的存数最多只能到 234 个 10 进位的 12 位数, 就已能用以证明 Simson 线那样的定理. 随着超大规模集成电路与其他技术的出现与改进, 微型机将愈来愈小型化而内存却愈来愈大, 功能愈来愈多, 自动化的程度也愈来愈高. 进入 21 世纪以后, 这一类方便的小型机器将为广大群众普遍使用. 它们不仅将成为证明一些不很简单的定理的武器, 而且还可用以发现并证明一些艰深的定理, 而这种定理的发现与证明, 在数学研究手工业式的过去, 将是不可想像的. 这里我们应该着重指出, 我们并不鼓励以后人们将使用计算机来证明甚至发现一些有趣的几何定理. 恰恰相反, 我们希望人们不再从事这种虽然有趣却即是对数学甚至几何学本身也已意义不大的工作, 而把自己从这种工作中解放出来, 把自己的聪明才智与创造能力贯注到更有意义的脑力劳动上去.

还应该指出, 目前我们所能证明的定理, 局限于已经发现的机械化方法的范围, 例如初等几何与初等微分几何之内. 而如何超出与扩大这些机械化的范围, 则是今后需要探索的长期的理论性工作.

4. 历史的启示与中国古代数学

我们发现几何定理证明的机械化方法是在 1976 至 1977 年之间. 约在两年之后我们发现早在 1899 年出版的 Hilbert 的经典名著《几何基础》中, 就有着一条真正的正面的机械化定理：初等几何中只涉及从属与平行关系的定理证明可以机械化. 当然, 原来的叙述并不是以机械化的语言来表达的, 也许就连 Hilbert 本人也并没有对这一定理的机械化意义有明确的认识, 自然更不见得有其他人提到过这一定理的机械化内容. Hilbert 是以公理化的典范而著称于世的, 但我认为, 该书更重要处, 是在于提供了一条从公理化出发, 通过代数化以到达机械化的道路. 自然, 处于 Hilbert 以及其后数学的一张纸一支笔的手工作业时代里, 公理化的思想与方法得到足够的重视与充分的发展, 而机械化的方向与意义受到数学家的忽视是完全可以理解的. 但电子计算机已日益普及, 因而繁琐而重复的计算已成为不足道的事情, 机械化的思想应比公理化思想受到更大重视, 似乎是合乎实际的.

其次应该着重指出, 我们从事机械化定理证明工作获得成果之前, 对 Tarski 的已有工作并无接触, 更没有想到 Hilbert 的《几何基础》会与机械化有任何关系. 我

们是在中国古代数学的启发之下提出问题并想出解决办法来的.

说起来道理也很简单：中国的古代数学基本上是一种机械化的数学. 四则运算与开方的机械化算法由来已久. 汉初完成的《九章算术》中, 对开平、立方与解线性联立方程组的机械化过程, 都有详细说明. 宋代更发展到高次代数方程求数值解的机械化算法.

总之, 各个数学领域都有定理证明的问题, 并不限于初等几何或微分几何. 这种定理证明肇始于古希腊的 Euclid 传统, 现已成为近代纯粹数学或核心数学的主流. 与之相异, 中国的古代学者重视的是各种问题特别是来自实际要求的具体问题的解决. 各种问题的已知数据与要求的数据之间, 很自然地往往以多项式方程的形式出现. 因之, 多项式方程的求解问题, 也就自然成为中国古代数学家研究的中心问题. 从秦汉以来, 所研究的方程由简到繁, 不断有所前进, 有所创新. 到宋元时期, 更出现了一个思想与方法的飞跃：天元术的创立.

“天元术”到元代朱世杰时又发展成四元术, 所引入的天元、地元、人元、物元实际上相当于近代的未知元或未知数. 将这些未知元作为通常的已知数那样加减乘除, 就可得到与近代多项式与有理函数相当的概念与相应的表达形式与运算法则. 一些几何性质与关系很容易转化成这种多项式或有理函数的形式及其关系. 这使得过去依题意列方程这种无法可循需要高度技巧的工作从此变成轻而易举. 朱世杰 1303 年的《四元玉鉴》又给出了解任意多至四个未知元的多项式方程组的方法. 这里限于 4 个未知元只是由于所使用的计算工具 (算筹和算板) 的限制. 实质上他解方程的思想路线与方法完全可以适用于任意多的未知元.

不问可知, 在当时的具体条件下, 宋世杰的方法有许多缺陷. 首先, 当时还没有复数的概念, 因之宋世杰往往限于求出 (正) 实值. 这无可厚非, 甚至在 17 世纪 Descartes 的时代也还往往如此. 但此外宋世杰在方法上也未臻完善. 尽管如此, 宋世杰的思想路线与方法步骤是完全正确的, 我们在 20 世纪 70 年代之末, 遵循宋世杰的思想与方法的基本实质, 采用美国数学家 J. F. Ritt 在 1932, 1950 年关于微分方程代数研究书中所提供的某些技术, 得出了解任意复多项式方程组的一般算法, 并给出了全部复数解的具体表达形式. 此后又得出了实系数时求实解的方法, 为重要的优化问题提供了一个具体的方法.

由于多种问题往往自然导致多项式方程组的求解, 因而我们解方程的一般方法可被应用于形形式式的问题. 这些问题可以来自数学自身, 也可以来自其他自然科学或工程技术. 在本丛书的第一本书, 吴文俊的《数学机械化》一书中, 可以看到这些应用的实例. 在工程技术方面的应用, 在本丛书中已有高小山的《几何自动作图与智能 CAD》与陈发来和冯玉瑜的《代数曲面拼接》两本专著. 上述解多项式方程组的一般方法已推广至代微分方程的情形. 许多应用以及相应论著正在酝酿之中.

5. 未来的技术革命与时代的使命

宋元时代天元术与四元术的创造, 把许多问题特别是几何问题转化成代数方程与方程组的求解问题. 这一方法用于几何可称为几何的代数化.12 世纪的刘益将新法与“古法” 比较, 称“省功数倍”, 这可以说是减轻脑力劳动使数学走上机械化的道路的一项伟大的成就.

与天元术的创造相伴, 宋元时代的数学又引进了相当于现代多项式的概念, 建立了多项式的运算法则和消元法的有关代数工具, 使几何代数化的方法得到了有系统的发展, 见于宋元时代幸以保存至今的杨辉、李冶、朱世杰的许多著作之中. 几何的代数化是解析几何的前身, 这些创造使我国古代数学达到了又一个高峰. 可以说, 当时我国已到达了解析几何与微积分的大门, 具备了创立这些数学关键领域的条件, 但是各种原因使我们数学的雄伟步伐就在这些大门之前停顿下来. 几百年的停顿, 使我们这个古代的数学大国在近代变成了数学上的纯粹入超国家. 然而, 我国古代机械化与代数化的光辉思想和伟大成就是无法磨灭的. 本人关于数学机械化的研究工作, 就是在这些思想与成就启发之下的产物, 它是我国自《九章算术》以迄宋元时期数学的直接继承.

恩格斯曾经指出, 枪炮的出现消除了体力上的差别, 使中世纪的骑士阶级从此销声匿迹, 为欧洲从封建时代进入到资本主义时代准备了条件. 近年有些计算机科学家指出, 个人用计算机的出现, 其冲击作用可与枪炮的出现相比. 枪炮使人们在体力上难分强弱, 而个人用计算机将使人们在智力上难分聪明愚鲁. 又有人对数学的未来提出看法, 认为计算机的出现, 将使数学现在一张纸一支笔的方法, 在历史的长河中, 无异于石器时代的手工方法. 今天的数学家们, 不得不面对计算机的挑战, 但是, 也不必妄自菲薄. 大量繁复的事情交给计算机去做了, 人脑将仍然从事富有创造性的劳动.

我国在体力劳动的机械化革命中曾经掉队, 以致造成现在的落后状态. 在当前新的一场脑力劳动的机械化革命中, 我们不能重蹈覆辙. 数学是一种典型的脑力劳动, 它的机械化有着许多其他类型脑力劳动所不及的有利条件. 它的发扬与实现对我国的数学家是一种时代的使命. 我国古代数学的光辉, 鼓舞着我们为实现数学的机械化, 在某种意义上也可以说是真正的现代化而勇往直前.

吴文俊
2002 年 6 月于北京

序　言

我们正处在信息时代, 电子计算机已经渗透到几乎所有的行业, 极大地提高了社会生产力、推动了社会的发展. 可以认为计算机是人脑的延伸, 电子计算机的飞速发展, 为人类实现脑力劳动的机械化创造了物质条件. 部分实现脑力劳动机械化, 将为科学研究与高新技术创新提供有力工具, 使科研工作者摆脱繁琐的甚至是人力难以胜任的工作, 将自己的聪明才智集中到更高层次的创新性研究上.

实现数学的机械化是实现脑力劳动机械化的重要理论基础. 数学是对现实世界数与形的最简洁、最高效、最优美的描述, 也有人把数学描述为具有 "内部抽象性和外部有效性的科学". 数学为其他学科提供了定量化描述问题的语言与解决问题的有效方法, 是科学技术的重要理论基础. 正是由于数学的这种基础性, 每个时代都有与之相适应的数学. 为利用计算机的强大计算能力, 数学的很大一部分内容正在转变为计算机可以接受的语言. 具体讲就是数学的离散化、算法化与软件化. 这样的数学也可以称之为机械化数学. 正是在此背景下, 吴文俊院士在 20 世纪 70 年代末提出了数学机械化的设想, 可以概括为如下的 "数学机械化纲领":

- 在数学的各个学科选择适当的范围实现机械化, 推动数学发展与脑力劳动机械化.
- 应用数学机械化方法解决相关高科技领域的关键基础理论问题.

现阶段数学机械化研究主要关注的是科学研究与高技术应用中经常遇到的两类问题: 方程求解与自动推理. 科学研究与高新技术研究中很多问题往往可以转化为各种方程的求解. 17 世纪的思想家 Rene Descartes 曾提出著名的 "Descartes 构想", 设想将任意问题的解答化为数学问题的解答, 将任意数学问题的解答化为代数问题的解答, 而代数问题则可以化为方程组求解. 这一设想虽然不能适用于所有问题, 却能够适用于很多非常重要的问题. 事实上, 方程求解已经成为当今诸多重要科学领域与高新技术研究的核心问题. 数学机械化方法的核心是方程求解的 "吴特征列方法". 这一方法不仅适用于代数方程, 而且适用于微分方程、差分方程、不等式系统. 脑力劳动的主要特点之一是推理功能. 计算机产生智能行为的一个关键在于实现推理的自动化. 自动推理是计算机产生后人们最先考虑的应用之一. 吴文俊院士于 70 年代末, 从中国传统数学中的机械化思想出发, 创立了几何定理机器证明的 "吴方法", 实现了几何自动推理研究的突破. 在吴文俊开创性工作的影响下, 经过 20 余年的努力, 已经形成具有我国特色的数学机械化和推理自动化理论, 成功地解决了一系列高新技术领域中的关键基础理论问题.

软件开发是数学机械化研究的有机组成部分. 数学机械化软件不仅可以用来验证相应算法的有效性, 更重要的是, 这些软件为数学机械化的应用提供了直接的工具. 数学机械化软件的开发走过了漫长的路程. 20 世纪 80 年代初期, 吴文俊编写的 “China-Prover” 首次实现了几何定理的机器证明. 此后 20 余年, 我们在数学机械化的主要方面包括: 方程求解、机器证明、几何作图、Ore 多项式、混合计算等形成了多个成功的软件. 但是其中大部分都是在其他软件, 如 Maple 的基础上开发的, 还没有一个基于我们自己理论的, 真正独立的通用软件平台. 这在很大程度上影响了数学机械化理论研究特别是应用的进一步发展. 数学机械化平台 MMP 就是在这样的背景下开发的.

基于上述考虑, 我们在 1998 年立项的国家基础研究发展规划项目 “数学机械化与自动推理平台” 中设立了软件开发课题组, 专门从事 MMP 的开发. MMP 的基本设想是使用 C++ 语言, 开发独立、完整、高效、具有自主知识产权的数学机械化软件平台, 为科学研究、工程应用、教学中的计算与推理提供了一个强有力的应用和开发平台. 经过课题承担人的共同努力, 我们于 2001 年底推出了自动推理平台 MMP-1.0 版本, 2003 年推出了完整的自动推理平台 MMP-2.0 版本. 此外, 我们还建立了 MMP 的网站①, 收录了 MMP 的使用方法与大量的测试问题. MMP 不仅实现了数学机械化的基本内容, 还包含了 “数学机械化与自动推理平台” 项目产生的很多最新成果. 目前 MMP 主要功能如下:

- 支撑系统, 包括内存管理、图形界面、编程环境.
- 符号计算基本运算系统, 包括任意长度的数系统, 多项式运算, 符号线性代数.
- 核心模块, 包括多项式方程、常微分方程、偏微分方程系统的吴特征列方法与投影定理.
- 应用模块, 包括:
  - ◇ MMP/Geometer: 几何定理自动证明与发现, 几何自动作图
  - ◇ MMP/DiffEquation: 微分方程求解
  - ◇ MMP/Identity: 组合恒等式自动证明
  - ◇ MMP/Blending: 过渡曲面自动生成
  - ◇ MMP/6R-Robots: 6R 机器人模拟

本书主要包括两部分内容: 一是对 MMP 的基本功能的介绍, 主要是前三章; 二是通过 MMP 的运行实例介绍数学机械化的基本理论与最新进展, 特别是在方程求解与机器证明方面的结果. 具体讲, 第四章介绍了多项式方程系统、常微分方程系统、偏微分方程系统的吴特征列方法与投影定理. 第五章介绍初等与微分几何中定

① http://mmrc.iss.ac.cn/ mmp.

理自动证明与自动发现的吴方法与若干最新进展，包括: 非退化条件的充分性、初等几何与微分几何公式的自动发现. 第六章介绍代数方程求解的吴特征列方法以及参数方程求解、预解式理论及其在机器人、曲面拼接、代数簇隐式化中的应用. 第七章介绍微分方程求解的吴特征列方法以及微分方程初等函数解、行波解、幂级数解的求解算法. 第八章介绍代数系统全局优化的吴有限核定理，不等式的自动证明与发现以及若干数值优化算法. 每章末还对相应各章的内容与 MMP 实现的方法所涉及的文献进行了介绍. 为了帮助读者熟悉 MMP, 本书收录了大量的例子, 并且都在 MMP 上实现. 所以, 本书既可以作为 MMP 的使用手册, 又可以作为数学机械化基本方法与最新成果的介绍.

MMP 的开发历时 5 年, 先后参加这一工作的人员包括: 高小山、王定康、裘宗燕、杨宏、林东岱、廖启征、王天明、武永卫、陈雪峰、侯春旺、朱长才、林强、程进三、张挺、龙红亮、赵新超、罗勇、吕卓生、冯如勇、王新民. 特别地, §8.4 由侯春旺完成. 在此对他们致予诚挚感谢. 由于 MMP 体系庞大, 又经过多位开发者长期的修改, 不可避免地存在问题. 高小山、王定康、裘宗燕、杨宏于 2005 年对 MMP 进行了全面整理、修改, 并最后定稿. MMP 还可能有不完善之处, 我们将对此负责, 并希望在以后的使用中不断改进.

MMP 是在 “973” 项目 “数学机械化与自动推理平台” 的直接支持下开发的. “数学机械化应用推广专项经费” 提供了部分支持. 在此, 我们对国家科技部、中国科学院与国家基金委对于数学机械化研究与软件开发给予的长期大力支持表示衷心感谢.

最后, 我们要感谢吴文俊院士对于 MMP 开发的长期支持与鼓励. 吴文俊院士在程序开发方面自己身先士卒, 年逾花甲还从头开始学习程序设计, 用 FORTRAN 语言写成了包括符号计算基本功能的几何定理证明系统. 正是在吴文俊院士这种精神鼓励下, 我们才得以克服种种困难, 在软件开发的道路上走到今天.

高小山

2006 年 6 月

目　　录

第一章　数学机械化平台 MMP 简介

我们将对数学机械化平台 MMP 做一个简要的介绍. 首先介绍 MMP 的安装与启动, 然后简单描述 MMP 的操作界面, 引导读者快速地了解 MMP 的用户环境. 通过一系列简单且有代表性的运行实例, 使读者对 MMP 的基本功能和使用方法有一个整体的印象.

§1.1　MMP　简　介

MMP 是一个基于 C++ 语言在微软 Windows 界面下开发的数学软件. 其核心功能是吴文俊发展的数学机械化基本理论, 包括多项式方程系统、代数常微分方程系统、代数偏微分系统方程的特征列方法与投影定理. 为了实现这些功能, MMP 还包含一个符号计算基本运算系统作为支撑部分. 此外, 作为数学机械化方法的应用, 我们还实现了方程求解、机器证明的各种方法. MMP 的程序、用户手册、测试例子可以在系统的网站上得到：

http://www.mmrc.iss.ac.cn/ mmp

MMP 由支撑系统、符号计算系统、核心模块与应用模块四个部分组成. 下面分别介绍各个模块的主要功能.

1. 支撑系统.　包括内存管理、图形界面、编程环境.

- *内存管理*　符号系统的一个基本特征是需要使用大量的计算机内存. 一个好的内存管理是符号系统成功的关键. MMP 实现了一个定长与一个变长的内存管理系统, 分别用于多项式与大整数的内部表示.
- *图形界面*　MMP 实现了图形用户界面系统, 使得系统界面友好, 易于使用. 用户可以通过图形界面方便地进行命令的输入、编辑和执行, 图形界面系统以图形方式显示命令的输出结果. 对于系统的输出结果, 用户可以进行选择、拷贝和粘贴操作.
- *编程环境*　编程语言解释环境是连接系统与用户的桥梁：MMP 通过语言解释器来理解用户的输入与要求. 作为一个数学软件, MMP 实现了一个可编程的语言解释环境. 该编程语言为用户提供了诸如赋值语句、循环语句、控制语句、过程定义等丰富的功能, 使用户能够在 MMP 中编写自己的函数, 实现复杂的算法.

2. 符号计算系统. 包括任意精度的数系统、多项式运算、符号线性代数.

- *数系统* 数运算系统是符号计算软件系统实现中最基本的组成部分, 是实现其他数学对象与算法的基础. MMP 提供了一个快速高效的多精度数运算系统, 实现了整数、小数、分数任意精度下的运算. 也就是说在 MMP 提供的数系统中, 只要系统的内存容许, 整数可以任意大, 小数运算可以达到任意精度. MMP 系统平台除了可以进行数的算术运算外, 还提供了一系列可供用户使用的系统函数. 如 GCD、素数分解、对数、中国剩余定理等.
- *有限域上的运算* 有限域上的运算包括有限域上数的运算与有限域上多项式的运算. MMP 系统平台提供了一系列有限域上有关数与多项式的运算操作, 例如有限域中元素平方根计算、有限域中元素逆元的计算、有限域上多项式的 GCD, 伪除、无平方分解、因式分解等.
- *多项式运算* 多变元多项式是符号计算软件系统的基本数据对象. MMP 所要实现的核心方法即数学机械化方法所要处理的基本对象就是多项式. MMP 实现了多项式的基本运算：各种算术运算以及多项式最大公因子计算的模算法、多项式因式分解的模算法等.
- *线性代数* 在线性代数方面, MMP 系统平台实现了符号矩阵运算、行列式计算、特征值计算以及线性方程组求解等功能.

3. 核心模块. 吴特征列方法是数学机械化方法的核心, 是几何定理机器证明、代数方程组求解以及其他应用模块的基础. MMP 系统平台目前实现了代数、常微、偏微情形的吴特征列方法与投影算法, 其中包括特征列序列的计算、代数扩域上的 GCD、代数扩域上的因式分解、三角列的计算等一系列关键算法. 为了提高效率, 我们实现了各种消去法, 包括因式分解、分支控制、子结式法、Seidenberg 法等. 所产生的特征列可以是 Ritt 意义下的特征列、吴意义下的特征列、弱特征列或正则列等.

4. 应用模块.

- MMP/Geometer 自 20 世纪 70 年代吴文俊发明几何定理机器证明的吴方法以来, 几何自动推理逐渐成为自动推理领域最活跃与成功的方向之一. MMP/Geometer 试图将几何推理有代表性的方法有机结合起来, 形成一个功能强大的几何问题求解系统. 这一系统实现了几何定理证明的吴方法、演绎数据库方法、面积法与几何图形自动生成的 C 树分解法、几何变换法与基于优化的数值算法. 使用本系统不仅可以证明定理还可以自动发现定理.
- MMP/Blending 本软件实现了给定的任意两个闭合二次曲面 (椭圆柱面、椭圆锥面、椭球面、单叶双曲面、双叶双曲面、椭圆抛物面) 轴心线共面和异面的

拼接.

- MMP/6R-Robots 空间 6R 机器人是一种一般的串连机器人. 6R 机器人的位置反解是在已知机器人的位置和姿态的情况下, 求解各个运动副的位移量. 这个问题本质上是多元非线性方程组的消元问题. 采用计算机仿真技术可以把机器人的工作情况形象显示出来, 从而可以方便地判断哪些解能够使机器人的运动达到要求, 并使运动过程不产生干涉. 本系统实现的功能包括: (1) 通过人机交互方式对结构参数和位置参数进行自由的设置和修改, 以适应各种不同系列的机器人, 达到尺寸链驱动的目的. (2) 对于每一个位置, 系统将进行反解计算, 得到所有的解. 使操作者能够对所有可能的解进行选择. (3) 对机器人进行运动仿真, 使设计者能够清楚地看到 6R 机器人的结构及外形, 以及机器人末端的运动轨迹. 从而可以方便地选择分支, 躲避障碍. (4) 可以根据已经设置的数据对机器人运动进行直线插补运算, 并显示其插补运动轨迹.
- MMP/DiffEquation 这一模块实现了微分方程求解功能. 对于一个微分方程组, 我们首先通过吴消元法将其归结为单个微分方程的求解. 对于单个微分方程, 我们可以求其有理函数解与代数函数解. 对于非线性 PDE, 我们还可以求其某种形式的行波解与孤立子解. 如果上述封闭形式解不存在, 我们可以求其形式幂级数解、Puiseux 幂级数解或 Adomian 幂级数解.
- MMP/Identity 这一模块实现了非交换差分算子的消去法, 并以此为基础编制了自动证明组合恒等式的程序.

本书将重点介绍 MMP 的方程求解与机器证明功能, 包括代数与微分方程的符号求解, 初等几何与微分几何中的定理证明. 我们还将介绍 MMP 方程求解与机器证明功能的若干应用, 包括在机器人、计算机辅助几何设计、计算机视觉等领域中的应用.

§1.2 MMP 的安装与启动

1. MMP 的安装与启动. MMP 可以在任何安装了 Microsoft 的 Windows 操作系统的微机上运行. 目前流行的 Windows 操作系统的各种版本均可支持 MMP 的运行, 例如 Windows 9x, Windows 2000 或 Windows XP 等.

首先把存放有 MMP 的光盘插入光盘驱动器中, 通过 Windows 的文件管理器, 用户可以在光盘中 MMP 的相关目录下找到安装程序 “setup. exe”, 直接用鼠标双击这个安装程序, 即可安装 MMP. 用户也可以从 Windows 窗口左下角的 “开始” 菜单中选择 “运行” 选项, 通过在打开的运行窗口中键入 “setup. exe” 运行安装程序, 完成 MMP 的安装. 在安装过程中, 用户可以指定 MMP 在硬盘上的安装目录. MMP

安装完成后, 将自动在 Windows 的桌面上产生 MMP 的图标, 用户只需用鼠标双击该图标即可启动 MMP. MMP 软件也可以从以下网址 http://mmrc.iss.ac.cn/mmp 免费下载.

2. MMP 的图形用户界面. 启动 MMP 之后, 其图形用户界面如图1.1 所示.

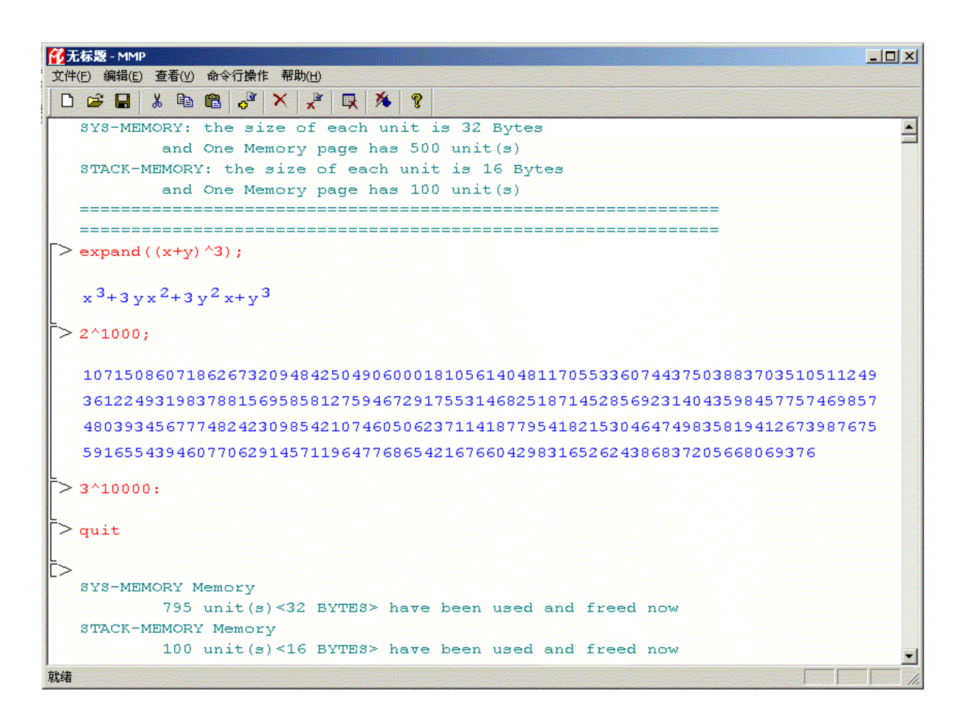


图 1.1　MMP 的图形界面

MMP 的窗口分为几个区域, 从上到下分别是标题栏, 菜单栏, 工具栏, 工作窗口和状态栏, 这些都是标准的窗口元素, 其中工具栏中的按钮将为用户提供最常用命令的快捷操作, 目前有文件的存取操作, 命令行的剪切, 复制与粘贴, 命令行的插入与删除等. 由于用户在 MMP 上的绝大多数操作都是在工作窗口上完成的, 运算结果也将呈现于工作窗口. 因此, 以下我们将重点介绍 MMP 的工作窗口.

启动 MMP 后, MMP 即自动打开一个新的工作窗口. 每个工作窗口中的内容可以存储为一个特殊的文件, 文件名后缀为 ".mm". 新建文件的默认文件名为 "无标题.mm". 用户可以在存储文件时按自己的意愿指定一个文件名.

在 MMP 的工作窗口上会自动产生提示符号 ">", 表示当前输入命令的位置. 输入的表达式必须用分号 ";" 或冒号 "：" 结尾, 以表示命令的结束. 然后按 Enter 键执行这个命令. 以 ";" 结尾表示显示运算结果, "：" 则表示不显示运算结果. 例如在上面的工作窗口中, 我们用整数求幂运算分别计算 2^{1000} 和 3^{10000}.

在命令行键入 “quit” 命令将结束一个工作窗口内的交互进程. 此时, MMP 会自动显示此次交互进程中系统内存使用情况的相关信息, 再按 Enter 键将关闭该工作窗口.

有关 MMP 的工作窗口, 除上述基本操作要点外, 还有以下几点值得注意:

- 目前 MMP 支持的输入输出方式分别为一维纯文本输入和二维平面输出.
- 工作窗口上出现的方括号为命令行括号, 它会自动将输入与所对应的输出括起来. 输入用红色文字显示, 输出则用蓝色文字显示.
- 以 # 开头的命令行将不被执行, 可以作为注解行使用.
- 工作窗口开头的几行绿色文字给出了系统内存设置的有关信息.
- 命令行的各种插入和删除操作可以通过选择 “命令行操作” 菜单下的相应选项或按一下工具栏中的相应按钮即可.
- 如果想知道工具栏上每一个按钮的功能, 只需要把鼠标指针移到按钮上, 按钮旁会出现该按钮的功能说明.

用户可以通过图形界面方便地进行命令的输入、编辑和执行. 图形界面系统以图形方式显示命令的输出结果. 对于系统的输出结果, 用户可以进行选择、拷贝和粘贴操作. 进行粘贴操作时, 系统自动将图形化的公式还原成其原始的输入格式. 用户还可以通过图形用户界面系统对所有执行过的命令行进行管理. 用户可以在任何位置插入一个新的命令行, 也可以删除一个命令行或者删除全部命令行, 还可以只删除一个命令行的输出结果或所有命令行的输出结果. 用户可以将执行过的命令行及其输出结果保存成 “.mm” 文件, 以备后用.

下面看几个例子:

```
>x*y^2+x/y;
```

$$\frac{x}{y}+xy^2$$

在上例中, 计算结果以图形方式显示. MMP 本身支持的是图形方式的输出, 但是如果把该图形方式的输出复制到文本中, 将自动变成其原始的纯文本格式.

```
>x*y^2+x/y;
 x/y+x*y^2
```

为方便起见, 我们经常会把图形输出直接复制到文本中. 因此读者会在本书中看到两种输出方式并存. 实际上, MMP 都是以图形方式显示输出结果的.

```
>(a+b)/(1+a/(1+a/b));
```

$$\frac{a+b}{1+\frac{a}{1+\frac{a}{b}}}$$

```
>(1+x)^(a+b);
```

$(x+1)^{(b+a)}$

```
>sqrt(x+y/(x+y));
```

$$\sqrt{\frac{y}{(y+x)}+x}$$

```
>matrix(3,3,[1/2,3,0,x^y,(1+x)/(x+y),0,(x+y)^100,1,sqrt(x)]);
```

$$\begin{bmatrix} \frac{1}{2} & 3 & 0 \\ x^y & \frac{1+x}{x+y} & 0 \\ (x+y)^{100} & 1 & \sqrt{x} \end{bmatrix}$$

选择 MMP 菜单栏中的“文件/保存”选项, 可以将上例中所有的计算内容保存到一个文件中. 例如“exam.mm”. 以后再选择“文件/打开”选项, 就可以恢复保存的所有内容.

在接下来的三节中, 我们将通过若干具体的运行实例, 介绍 MMP 的使用方法与基本功能, 从而帮助用户能够快速掌握 MMP.

§1.3　数与多项式运算

本节将介绍 MMP 的数, 多项式与表达式运算.

1. 数的运算.　作为符号计算系统, MMP 具有处理无穷精度数值类型的能力. 只要计算机内存允许, MMP 可以处理任意大小、任意精度的数值. MMP 的数值类型包括整数、分数、浮点数和复数等类型. MMP 可以进行这些数值类型的运算, 以及它们的混合运算. 下面我们计算复数 $2+3I$ 的 200 次幂.

```
>n:=(2+3I)^200;
n:=-5149111069335940622488137480804322710027719408025413992998
7197169332830442740487455463163703057455821296640599+
2425277127345465276782435282153481737193222526521225279
6143294091510089491145789315484351881327354652732256740000*I
```

在本书的算例中, 符号 “>” 后面的公式为用户输入, 其余部分为计算结果.

MMP 中提供了关于素数的运算. 考虑第 6 个 Fermat 数：

```
>F:=2^(2^6)+1;
18446744073709551617
>isprime(F);
0
>ifactor(F)
```

```
[274177, 1, 67280421310721, 1]
>nextprime(F);
18446744073709551629
```

isprime(F) 判断 F 是否为素数. 返回 1 表示是素数, 返回 0 表示不是素数. 这里判断出 F 不是素数; ifactor(F) 给出 F 的素数分解; nextprime(F) 给出比 F 大的下一个素数.

在进行数的混合运算时, MMP 首先将各种类型转化为优先级最高的类型, 然后再进行运算.

```
>(2+3I)*23423.23423+2/3-20;
46827.1351266666666666666666667+70269.70269I
```

本例中复数类型的优先级最高, 因此最终运算结果是复数. 在 MMP 中浮点数的计算精度预设为 30 位有效数字, 这对于大多数的计算已经足够了.

考虑数的开方运算：

```
>nsqrt(12345^13);
```

$393271172272916707746482964$

```
>nsqrt(12345.^13);
```

$393271172272916707746482964.101$

```
>sqrt(12345^13);
```

$\sqrt{154662214940914131102165197707101295849230845947265625}$

```
>expand(sqrt(12345^13));
```

$353953788908662482314062 5\sqrt{12345}$

下面解释上述四条指令的意义.

- 第一条指令的参数是整数类型, 运算结果是 12345^{13} 的平方根的整数部分.
- 第二条指令把参数变成浮点数类型, 则该指令进行近似运算.
- 第三条指令要求求得 $\sqrt{12345^{13}}$ 的精确值, 此时不作任何化简, 因此结果仍然是原来的形式.
- 利用 expand 指令对 $\sqrt{12345^{13}}$ 进行化简, 得到化简后的精确结果.

2. 表达式与多项式. 在很多情形, 我们不需要将每个数学表达式展开. 为此, MMP 实现了表达式及其运算.

```
>p:=(3*x+7*y)^500;
>q:=(3*x+7*y)^499;
>p/q;
```

以上三个命令的输出分别是 $(7y+3x)^{500}$, $(7y+3x)^{499}$, $3x+7y$. 直接运算得到的表

达式一般是不展开的. 为了得到多项式, 我们需要对表达式进行展开.

```
>s:=expand(p):
>plength(s);
```

501

展开后我们得到一个 501 项的多项式. MMP 命令 plength 给出多项式的项数. 由于第一个命令的输出结果太长, 我们用冒号 "：" 结尾, 没有把结果显示出来.

多项式最大公因子与多项式因式分解的计算是符号计算的最基本的功能之一. 一般认为, 最大公因子与因式分解高效算法的出现标志着符号计算领域的开始. 我们在 MMP 中实现了最大公因子与因式分解计算的模算法.

```
>f1:=(73*x^2+3*y^2*z+z^2)^7*(79*u^2-8*v*w+3*u^3+67*v)^2:
>f2:=(68*x^2+3*y^2*z+z^2)*x^3*y^2*(73*x^2+3*y^2*z+z^2)^7*
     (79*u^2-8*v*w+3*u^3+67*v)^2:
>g:=gcd(f1, f2):
>factor(g);
```

$[[1], 1, [73x^2 + 3zy^2 + z^2], 7, [3u^3 + 79u^2 - 8wv + 67v], 2]$

```
>factor(f2);
```

$[[1], 1, [x], 3, [y], 2, [73x^2+3zy^2+z^2], 7, [68x^2+3zy^2+z^2], 1, [3u^3+79u^2-8wv+67v], 2]$

这里 f_1 是一个 6 变元 27 次 360 项多项式, f_2 是一个 6 变元 35 次 450 项多项式. 由于最大公因子 (gcd) 的运算结果太长, 我们也没有显示出来.

在多项式运算中遇到数的混合运算时, 将遵循与数的混合运算同样的规则进行类型转化和精度设置. 例如：

```
>f:=expand((3*x+(2/7)*y+(8.0)*z+(2+3I)*w)^2);
```

得到如下结果：

$$9x^2 + \frac{12}{7}yx + 48.0zx + (12 + 18I)wx + \frac{4}{49}y^2 + 4.571428571428571428571428571$$
$$43zy + \left(\frac{8}{7} + \frac{12}{7}I\right)wy + 64.00z^2 + (32.0 + 48.0I)wz + (-5 + 12I)w^2.$$

3. 矩阵与线性方程组.　矩阵输入方式如下：

```
>B:=matrix(3, 3, [a, b, c, a^2, b^2, c^2, a^3, b^3, c^3]);
```

$$\begin{bmatrix} a & b & c \\ a^2 & b^2 & c^2 \\ a^3 & b^3 & c^3 \end{bmatrix}$$

```
>det(B);
```

$-cb^2a^3 + c^2ba^3 + cb^3a^2 - c^3ba^2 - c^2b^3a + c^3b^2a$

在 MMP 中矩阵的元素可以是数, 也可以是符号或其他 MMP 所支持的基本数据类型. 考虑下面的线性方程组

$$\begin{pmatrix} a & 2 & -7 \\ 4 & b & -18 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 5 \\ -33 \end{pmatrix}$$

即

$$\begin{cases} ax & +2y & -7z & -5 & = & 0 \\ 4x & +by & -18z & +33 & = & 0 \end{cases}$$

我们可以调用 MMP 的命令 lsolve 对以上方程组进行求解：

```
>a:=[a*x+2*y-7*z-5, 4*x+b*y-18*z+33]:
>b:=[x, y, z]:
>lsolve(a, b);
```

$$\left[\frac{(7bz-36z+5b+66)}{(ba-8)}, \frac{(18az-28z-33a-20)}{(ba-8)}, z\right]$$

以上输出结果代表的含义如下：

$$\begin{cases} x = \dfrac{7zb+5b-36z+66}{ba-8} \\ y = \dfrac{18za-33a-28z-20}{ba-8} \\ z = z \end{cases}$$

z 是自由变量.

§1.4　用 MMP 求解代数与微分方程

1. 代数方程求解.　考虑下面所谓“4 循环问题”.

$$\begin{aligned} & x_4+x_3+x_2+x_1=0 \\ & x_4*x_3+x_3*x_2+x_2*x_1+x_1*x_4=0 \\ & x_4*x_3*x_2+x_3*x_2*x_1+x_2*x_1*x_4+x_1*x_4*x_3=0 \\ & x_4*x_3*x_2*x_1-1=0 \end{aligned}$$

这是一个代数方程组. 从这个方程组本身, 我们并不能看出其解的性质. 例如, 这个方程组的解是否有限？有几个解？等等. 用吴消元法将上述方程组化简, 即可回答这些问题. 在 MMP 中, 可以用 wsolve 函数实现这一化简功能.

```
>wsolve([x4+x3+x2+x1, x4*x3+x3*x2+x2*x1+x1*x4,
        x4*x3*x2+x3*x2*x1+x2*x1*x4+x1*x4*x3, x4*x3*x2*x1-1],
        [x4, x3, x2, x1]);
Time is: 0.453 second(s).
[[x1*x4-1, x3+x1, x1*x2+1], [x1*x4+1, x3+x1, x1*x2-1]]
```

由上述计算我们知道, 原方程组的零点等价于下面方程组的零点.

$$x_1x_4 - 1 = 0, \quad x_3 + x_1 = 0, \quad x_1x_2 + 1 = 0$$
$$x_1x_4 + 1 = 0, \quad x_3 + x_1 = 0, \quad x_1x_2 - 1 = 0$$

由于上面方程组的特殊“三角形”结构, 我们容易知道原方程组的解是一维的代数簇. 对于一个非零的 $x_1 = a$, 其他变量可以求解如下 $x_4 = \frac{1}{a}, x_3 = -a, x_2 = -\frac{1}{a}$ 或 $x_4 = -\frac{1}{a}, x_3 = -a, x_2 = \frac{1}{a}$.

对于简单的问题, MMP 直接给出其精确解.

```
>roots([x^2+(p-10)*x-9*p+9, y^3-x+1], [x, y, p]);
```

$$\Big[[y, -I\sqrt{3}-1, x, 9], [p], [y, I\sqrt{3}-1, x, 9], [p], [y, 2, x, 9], [p], [y, -\sqrt[3]{p}, x, -p+1], [p],$$
$$\Big[y, \frac{1}{2}I\sqrt[3]{p}\sqrt{3}+\frac{1}{2}\sqrt[3]{p}, x, -p+1\Big], [p], \Big[y, -\frac{1}{2}I\sqrt[3]{p}\sqrt{3}+\frac{1}{2}\sqrt[3]{p}, x, -p+1\Big], [p]\Big]$$

在本例的输出结果中, p 代表自由变元. 该输出结果给出方程组 $x^2+(p-10)x-9p+9=0, y^3-x+1=0$ 的精确解如下：

$$y = 2, x = 9$$
$$y = -\sqrt{3}p, x = -p+1$$
$$y = \pm I\sqrt{3}-1, x = 9$$
$$y = \pm\frac{1}{2}I\sqrt[3]{p}\sqrt{3}+\frac{1}{2}\sqrt[3]{p}, x = -p+1$$

如果方程组的解不能用根号精确表示, MMP 可以对方程组的实根进行实根隔离. 考虑下面的例子：

```
>Mrealroot([x-2, y^2+x-6, z^2+x+y-10], [x, y, z], 1/1000);
[[[2,2],[-2,-2],[-3239/1024,-1619/512]],
[[2,2],[-2,-2],[1619/512,3239/1024]],
[[2,2],[2,2],[-2509/1024,-627/256]],
[[2,2],[2,2],[627/256,2509/1024]]]
```

上述计算结果表明, 我们考虑的方程系统有四组实根, 分别位于上面的四组区间内. 例如, 第二组实根是 $x = 2, y = -2, z \in (\frac{1619}{512}, \frac{3239}{1024})$, 这里 x, y 的值是精确的. 通过将第三个参数设为更小的数, 可以得到 z 所属的更小的区间.

```
>Mrealroot([x-2, y^2+x-6, z^2+x+y-10], [x, y, z], 1/100000);
[[[2,2],[2,2],[-80265/32768,-321059/131072]],
[[2,2],[2,2],[321059/131072,80265/32768]],
[[2,2],[-2,-2],[-414487/131072,-207243/65536]],
[[2,2],[-2,-2],[207243/65536,414487/131072]]]
```

则得到上面考虑的实根的更准确的近似值：$x=2, y=-2, z\in\left(\frac{207243}{65536},\frac{414487}{131072}\right)$.

对于很多的应用, 我们需要给出方程组的近似解, 而不是解所属的区间. 这时我们可以调用 MMP 的函数 pssolver. 函数 pssolver 可以给出方程组在复数域上的所有近似解. 考虑同一个例子.

```
>pssolver([x-2, y^2+x-6, z^2+x+y-10], [x, y, z]);
[[2,2,2.44949],[2,2,-2.44949],[2,-2,3.16228],[2,-2,-3.16228]]
```

函数 pssolver 给出了方程组的四组实根的近似解. 例如, 近似解 $x=2, y=-2, z=3.16228$ 中 z 的值落在函数 Mrealroot 所给出的区间内.

最后, 我们考虑一个机器人问题. 考虑图1.2所示的一个广义 Stewart 平台. 这个广义 Stewart 平台的动平台与静平台分别是包含直线 l_1, l_2 和 l_3, l_4 的两个刚体. 动平台与静平台之间由三个角度约束与三个距离约束连接：

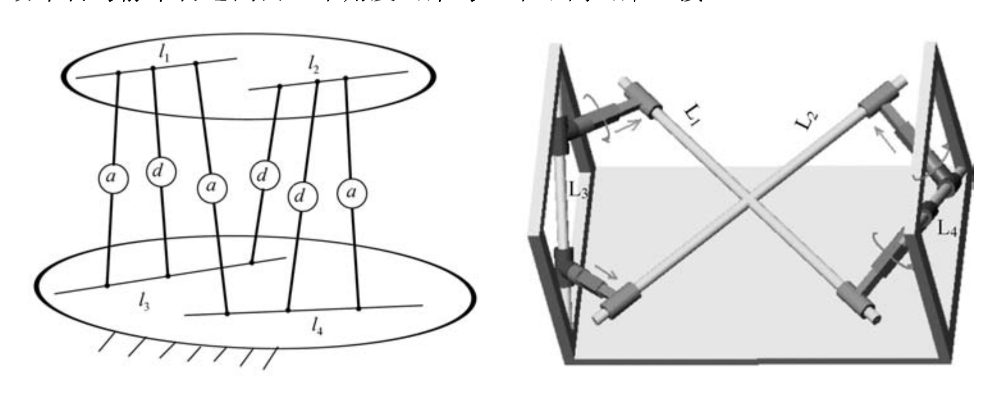


图 1.2 一个特殊的广义 Stewart 平台

$$\cos(\angle(l_1,l_3))=d_1, \cos(\angle(l_1,l_4))=d_2, \cos(\angle(l_2,l_4))=d_3 \tag{1.1}$$

$$\mathrm{DIS}(l_1,l_3)=r_1, \mathrm{DIS}(l_2,l_3)=r_2, \mathrm{DIS}(l_2,l_4)=r_3 \tag{1.2}$$

所谓正解问题是指由上述六个约束确定 l_1, l_2 所在动平台的位置. 我们首先用 (1.1) 中的三个角度约束确定动平台的方向. 假设旋转矩阵是 $\mathbf{R}=(r_{ij})_{3\times3}$, 直线 l_i 的单位方向向量为 $\mathbf{s}_i (i=1,2,3,4)$, 则角度约束是

$$\begin{cases} \mathbf{R^TR}=\mathbf{I},\ \det(\mathbf{R})=1 \\ \mathbf{s_1}\cdot\mathbf{Rs_3}=d_1,\ \mathbf{s_1}\cdot\mathbf{Rs_4}=d_2,\ \mathbf{s_2}\cdot\mathbf{Rs_4}=d_3 \end{cases} \tag{1.3}$$

因为 l_i 是固定在基体和平台上的, 我们可以不失一般性地假定：

$$\mathbf{s_1} = (0,\, 0,\, 1),\, \mathbf{s_2} = (0, m_0, n_0),\, \mathbf{s_3} = (0, 0, 1)^T,\, \mathbf{s_4} = (0, m_1, n_1)^T$$

下面的运算可以得到关于 $\mathbf{R^TR} = \mathbf{I}$ 的 9 个方程与方程 $\det(\mathbf{R}) = 1$.

```
>R:=matrix(3, 3, [r11, r12, r13, r21, r22, r23, r31, r32, r33]);
>P:=multiply(tran(R), R);
>h1:=getval(P, 1, 1)-1;
>h2:=getval(P, 1, 2);
>h3:=getval(P, 1, 3);
>h4:=getval(P, 2, 1);
>h5:=getval(P, 2, 2)-1;
>h6:=getval(P, 2, 3);
>h7:=getval(P, 3, 1);
>h8:=getval(P, 3, 2);
>h9:=getval(P, 3, 3)-1;
>h10:=det(R)-1;
```

关于三个角度约束的方程可以如下得到：

```
>h11:=expand(getval(multiply(matrix([0, 0, 1], 1),
multiply(R, matrix([0, 0, 1], 0))), 1, 1)-d1);
>h12:=expand(getval(multiply(matrix([0, 0, 1], 1),
multiply(R, matrix([0, m1, n1], 0))), 1, 1)-d2);
>h13:=expand(getval(multiply(matrix([0, m0, n0], 1),
multiply(R, matrix([0, m1, n1], 0))), 1, 1)-d3);
```

对方程系统 $h_i = 0, i = 1, \cdots, 13$, 在变量序 $r_{32} < r_{31} < r_{23} < r_{22} < r_{21} < r_{13} < r_{12} < r_{11} < r_{33}$ 下, 利用吴特征列方法化为三角列形式如下：

```
>wsolve([h1,h2,h3,h4,h5,h6,h7,h8,h9,h10,h11,h12,h13],
        [r33, r11, r12, r13, r21, r22, r23, r31, r32],[],"weak");
```

由约化结果可以看到, 加入三个角度约束, 平台的方向有四个解.

现在考虑加入三个距离约束. 此时, 动平台的方向已经确定. 我们只需要确定动平台上的一点的位置, 就可以确定动平台的位置. 分别取定 l_1, l_2 上点 P_1, P_2. 因为 l_1, l_2 的方向已经确定. 再加上距离约束 $\mathrm{DIS}(l_1, l_3) = r_1$ 后, l_1 的轨迹是一个平面, 记做 p_1. 同样 l_2 在距离约束 $\mathrm{DIS}(l_2, l_3) = r_2$ 下的轨迹和在距离约束 $\mathrm{DIS}(l_2, l_4) = r_3$ 下的轨迹分别是一个平面, 记做 p_2, p_3. 因此, P_1 在平面 p_1 上, P_2 在平面 p_2 与 p_3 上. 又由于 l_1 与 l_2 的相对位置已知, P_2 还在将 p_1 沿着 P_1 到 P_2 的方向做平移所得

的平面 p_1' 上. 这样 P_2 就是三个平面 p_1', p_2, p_3 的交点. 因而, P_2 只有一个解. 因此这个广义 Stewart 平台的正解的个数是 4.

用 MMP 求解代数方程的详细描述请见第六章.

2. 微分方程求解. 考虑下面关于 x 的函数 $y(x)$ 的常微方程：

$$F = y'^3 + 4y'^2 + (27y^2+4)y' + 27y^4 + 4y^2 = 0$$

我们知道, 求非线性微分方程的精确解是非常困难的. 但是, 对于形如 F 的 *一阶常系数微分方程*, 我们可以用 MMP 快速求解其有理函数与代数函数通解. 使用 MMP, 可以得到 $F=0$ 的通解：$\hat{y} = \frac{(x+c)^2+1}{(x+c)^3}$, 其中 c 是任意常数. 这里用到的算法请见第七章.

```
>read("de_rat.txt");
>F:=y[1]^3+4*y[1]^2+(27*y[0]^2+4)*y[1]+27*y[0]^4+4*y[0]^2:
>ratsolve(F);
the rational solution is: (x^2+1)/(x^3)
```

作为特殊情况, ratsolve 也可以得到多项式解. 考虑下面常微方程：

$$F = y'^2 - 4y - 4 = 0$$

用 MMP 可以得到 $F=0$ 的通解：$\hat{y} = (x-c)^2 - 1$, 其中 c 是任意常数.

```
>read("de_rat.txt");
>F:=y[1]^2-4*y[0]-4:
>ratsolve(F);
the rational solution is: x^2-1
```

如果微分方程不存在有理解, 函数 ratsolve 也可以判断这一事实.

```
>read("de_rat.txt");
>F:=y[1]^2-y[0]^2-4:
>ratsolve(F);
no rational solution
```

我们知道, 大部分微分方程的解是不能够用初等函数表示的. 使用 MMP, 我们可以在微分方程解函数的非奇异点附近求其幂级数展开. 上面提到微分方程 $F = y'^2 - y^2 - 4 = 0$ 没有有理解. 我们可以用 MMP 求它在给定初值 $y(0)=1$ 下的幂级数解的前 5 项.

```
>read("de_taylora.txt");
>read("de_taylorb.txt");
```

```
>F:=y[1]^2-y[0]^2-4:
>taylorsolve(F,1,[1],5);
```

得到 $y = x*Z+1+1/2*x^2+1/6*Z*x^3+1/24*x^4+1/120*Z*x^5$, 其中 Z 满足 $Z^2-5=0$. 相似的, 求微分方程 $F = y'^2-4y = 0$ 在给定初值 $y(0) = 1$ 下的幂级数解的前 5 项.

```
>read("de_taylora.txt");
>read("de_taylorb.txt");
>F:=y[1]^2-4*y[0]:
>taylorsolve(F,1,[1],5);
```

得到 $y = x^2-2x+1$. 带入 $F = 0$, 可以验证是方程的解. 由此得到微分方程的多项式解.

对于一般的微分方程, 我们还可以用 MMP 分析其 Lie 对称群的生成方程组. 考虑下面关于 x 的函数 $y(x)$ 的常微方程：

$$y''(x) = ay(x)(y'(x)+by(x)) \tag{1.4}$$

其中 a, b 是任意常数. 方程 (1.4) 在变换 $(x, y) \longrightarrow (x+\xi\varepsilon+O(\varepsilon^2), y+\eta\varepsilon+O(\varepsilon^2))$ 下的无穷小 Lie 对称群的生成方程组是

$$\begin{aligned}
&\xi_{yy} = 0\\
&-2\,a\,y\,\xi_y+\eta_{yy}-2\,\xi_{xy} = 0\\
&-a\,\eta-3\,a\,b\,y^2\xi_y-a\,y\,\xi_x+2\,\eta_{xy}-\xi_{xx} = 0\\
&-2\,a\,b\,y\,\eta-2\,a\,b\,y^2\,\xi_x+a\,b\,y^2\,\eta_y-a\,y\,\eta_x+\eta_{xx} = 0
\end{aligned}$$

```
>depend([e, f], [x, y]);
>h1:=e[2, 0];
>h2:=-2*a*y*e[0, 1]+f[0, 2]-2*e[1, 1];
>h3:=-a*f - 3*a*b*y^2*e[0, 1]-a*y*e[1, 0]+2*f[1, 1]-
    e[2, 0];
>h4:=-2*a*b*y*f-2*a*b*y^2*e[1, 0]+a*b*y^2*f[0, 1]-
    a*y*f[1, 0]+f[2, 0];
>dwsolve([h1, h2, h3, h4], [e, f]);
```

通过运算可以得到生成方程组的如下解：

- $a \neq 0, b \neq 0$, $\xi = c_1, \eta = 0$.
- $a \neq 0, b = 0$, $\xi = c_1-c_2x, \eta = c_2y$.
- $a = 0, b = 0$, $\xi = c_1+c_2x+c_3y+c_4x^2+c_5xy, \eta = c_6+c_7x+c_8y+c_4xy+c_5y^2$.

用 MMP 求解微分方程的详细描述请见第七章.

§1.5　用 MMP 自动证明与发现定理

自动证明几何定理是 MMP 的基本功能之一. 考虑下面例子.

1. 定理自动证明.

例 1.5.1 (Simson 定理)　*D 是三角形 ABC 外接圆上一点. 证明由 D 向三角形 ABC 三边所作垂线之垂足共线*(图 1.3).

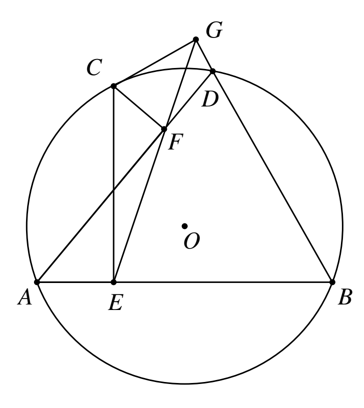


图 1.3　Simson 定理

MMP 提供了一种类似自然语言输入几何命题的方式. 例如, 在 MMP 中可以如下证明 Simson 定理.

```
>wprove("Example Simson. Let D be a point on the circumcircle O of
the triangle ABC. E is the foot from point D to line AB. F is the
foot from point D to line BC. G is the foot from point D to line
AC. Show that points E, F, and G are collinear.");
The geometric Statement is valid!
Time is: 375 microsecond.
```

这里, wprove 是 MMP 的一个函数, 它使用吴方法证明几何定理.

为了证明 Simson 定理, MMP 主要进行以下三步操作. 首先, MMP 将上述几何命题转换为谓词形式. 在 MMP 中这一步可以如下实现.

```
>geomtopd("Example Simson. Let D be a point on the circumcircle O
of the triangle ABC. E is the foot from point D to line AB. F is
the foot from point D to line BC. G is the foot from point D to
line AC. Show that points E,F,and G are collinear.");
```

```
[[y_G,x_G,y_F,x_F,y_E,x_E,y_D,x_D,y_O,x_O],
[v_C,u_C,v_B,u_B,v_A,u_A],[A,[0,0],B,[0,v_B],C,[u_C,v_C],
O,[x_O,y_O],D,[x_D,y_D], E,[x_E,y_E],F,[x_F,y_F],G,[x_G,y_G]],
[[point,A,B,C],[cong,O,A,O,B],[cong,O,A,O,C],[cong,O,D,O,A],
[coll,E,A,B],[perp,E,D,A,B],[coll,F,B,C],[perp,F,D,B,C],
[coll,G,A,C],[perp,G,D,A,C]],[[coll,A,B,C],[sqdis,A,B],
[sqdis,B,C],[sqdis,A,C]],[[coll,E,F,G]]];
```

Simson 定理的以上谓词表示的意义如下：

- y_G, x_G, v_C, u_C 是 MMP 引入的变量. 几何命题中的每个点都被赋予坐标：$A = [0,0], B = [0, v_B], C = [u_C, v_C]$ 等等.
- 几何命题中的条件用几何谓词表示. 例如 [cong, O, A, O, B] 表示 $OA = OB$, [coll, E, A, B] 表示 E, A, B 三点共线, [perp, E, D, A, B] 表示 $ED \perp AB$.
- 在列表 [[coll, A, B, C], [sqdis, A, B], [sqdis, B, C], [sqdis, A, C]] 中给出的几何条件是由 MMP 引入的几何非退化条件：A, B, C 不共线, $|AB| \neq 0$, $|BC| \neq 0$, $|AC| \neq 0$.
- 几何谓词 [coll, E, F, G] 是几何命题的结论.

接下来, MMP 将 Simson 定理的上述命题描述转变为代数形式.

```
>geom("Example Simson. Let D be a point on the circumcircle O of
the triangle ABC. E is the foot from point D to line AB. F is the
foot from point D to line BC. G is the foot from point D to line
AC. Show that points E, F, and G are collinear. ");
```

得到命题中几何条件的代数方程：

$$
\begin{aligned}
h_1 &= 2 * v_B * y_O - v_B^2 \\
h_2 &= 2 * v_C * y_O + 2 * u_C * x_O - v_C^2 - u_C^2 \\
h_3 &= y_D^2 - 2 * y_O * y_D + x_D^2 - 2 * x_O * x_D \\
h_4 &= -v_B * x_E \\
h_5 &= -v_B * y_E + v_B * y_D \\
h_6 &= u_C * y_F - v_C * x_F + v_B * x_F - v_B * u_C \\
h_7 &= -v_C * y_F + v_B * y_F - u_C * x_F + v_C * y_D - v_B * y_D + u_C * x_D \\
h_8 &= u_C * y_G - v_C * x_G \\
h_9 &= -v_C * y_G - u_C * x_G + v_C * y_D + u_C * x_D \\
d_1 &= -v_B * u_C \\
d_2 &= v_B^2
\end{aligned}
$$

$d_3 = v_C^2 - 2 * v_B * v_C + u_C^2 + v_B^2$
$d_4 = v_C^2 + u_C^2$
$c = x_F * y_G - x_E * y_G - y_F * x_G + y_E * x_G + x_E * y_F - y_E * x_F$

最后, MMP 使用关于代数方程的特征列方法证明：

$$[(h_1 = 0 \wedge \cdots \wedge h_9 = 0 \wedge d_1 \neq 0 \wedge d_2 \neq 0 \wedge d_3 \neq 0 \wedge d_4 \neq 0) \Rightarrow c = 0]$$

由于在实数情形, $|AB| = 0$, $|BC| = 0$, $|AC| = 0$ 是条件 "A, B, C 不共线" 的特殊情形, 我们实际上证明了 "如果 A, B, C 不共线, D 是三角形 ABC 外接圆上的一点, 则由 D 向三角形 ABC 三边所作垂线之垂足共线."

使用 MMP, 我们还可以自动证明微分几何与力学中的定理. 考虑下面问题.

例 1.5.2 由 Kepler 行星运动定理推导 Newton 定理.

Kepler 行星运动定理是

K1：行星的轨迹是以太阳为一个焦点的椭圆.

K2：从太阳出发指向行星的向量在相等时间扫过相等的面积.

Newton 万有引力定理是

N1：行星的加速度与太阳和行星之间的距离的平方成反比.

下面, 我们将由 K1, K2 证明 N1. 设太阳在坐标系的原点, 行星的坐标为 (x, y).

```
>depend([a, r, x, y, e, p], [t]);
>wdprove(
[[a, r, y, x, e, p], [], [],
  [r^2-x^2-y^2, (r是行星到太阳的距离)
   a^2-diff(x, t, 2)^2-diff(y, t, 2)^2, (a是行星的加速度)
   r - p - e * x, (行星的椭圆方程)
   diff(p, t), diff(e, t), (p, e是常数)
   x* diff(y, t, 2)-diff(x, t, 2)*y  (K2的微分方程表示)],
  [p], [diff(a*r^2, t)]]);
The geometric Statement is valid!
```

这里 depend 函数说明 a, r, y, x, e, p 是变量 t 的函数. 我们将椭圆的焦点取为原点, r 是行星到太阳的距离, a 是行星的加速度, $r - p - e * x = 0$ 是椭圆方程, $xy'' - x''y = 0$ 是 K2 的微分方程表示. 结论是 $a * r^2$ 是一个常数. 几何命题的非退化条件是 $p \neq 0$, 即轨迹不退化为直线. 运行 wdprove, 将代表问题假设的微分方程的解空间分解为七个分支, 并在每个分支上验证所述定理是正确的.

2. 定理自动发现. 使用 MMP, 我们不仅可以证明几何定理, 还可以自动发现几何

规律. 考虑下面问题.

例 1.5.3 (Peaucellier 连杆)　连杆 AD, AB, DC 和 BC 长度相等, 连杆 EA 和 EC 的长度相同. FD 的长度等于 E 到 F 的距离. 连接点 E 和 F 的位置为平面上的固定点, 连杆允许绕这些点旋转. 问连接点 B 的轨迹如何?(图 1.4)

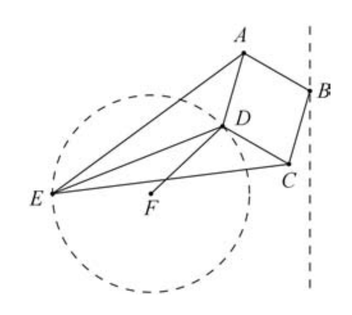


图 1.4　生成直线的 Peaucellier 连杆

令 $r = |FD|, |CB|^2 = |CD|^2 = n^2 + m^2$, 则 $|EC|^2 = (n+2r)^2+m^2$. 使用 MMP 的函数 wderive 可以自动推导 B 点的轨迹方程.

```
>wderive([[y2,x2,y1,x1,x],[r,m,n,y],
[F,[0,0],E,[r,0],C,[x2,y2],D,[x1,y1],B,[x,y]],
[[dis,F,D,r],[nminus,[dis,C,D],[nprod,n,n],[nprod,m,m]],
[nminus,[dis,C,B],[nprod,n,n],[nprod,m,m]],[nminus,[dis,E,C],
[npower,[nsum,n,r,r],2],[nprod,m,m]],[coll,E,D,B]],
[[nminus,x1,x]],[]]);
Formula 1 : x+r+2*n
```

由以上计算可知, 点 B 的轨迹是一条垂直于 EF 的直线: $x = -r - 2n$.

例 1.5.4　我们现在由 Kepler 定律自动推导出 Newton 引力定律.

在变量顺序 $e < p < x < y < r < a$ 下使用微分情形吴特征列方法:

```
>depend([a,r,x,y],[t]);
>vs:=[a,r,y,x,p,e];
>ps:=[r^2-x^2-y^2,a^2-x[2]^2-y[2]^2,r-p-e*x,x*y[2]-x[2]*y];
>ds:=[a,p];
>dwsolve(ps,vs,ds,"weak");
```

在上述运算中, 如果我们关注 a 和 r 的关系, 会得到微分方程: $a'r + 2ar' = 0$. 我们有 $a'r + 2ar' = \frac{(ar^2)'}{r}$. 所以 a 与 r^2 成反比. 这样, 在一定意义下, 我们由 Kepler 定律自动推导出 Newton 万有引力定律.

关于几何定理的自动证明与发现的详细介绍可见本书第五章.

第二章　MMP 的基本数据类型与运算

符号计算是数学机械化算法的重要支撑. MMP 具有完善的处理符号和数值运算的能力, 为建立其特有的自动推理功能打下良好的基础. 本章将详细介绍 MMP 的基本符号与数值运算功能.

§2.1　数 据 类 型

1. MMP 中的数据类型.　用户可以在 MMP 的界面上输入它所支持的各种数学对象, 这些数学对象将被 MMP 的解释系统识别为相应的数据类型. 本节将引导读者初步认识 MMP 的常用数据类型.

在 MMP 中, 可以用 type 指令查询对象的数据类型, type 返回该对象在 MMP 中内设的类型名. 例如:

```
>type(20);
Number(BigNumber)
>type(3.14);
Number(FloNumber)
```

在 MMP 中, 任何类型的数都首先被统一识别为基本数类型 (Number), 然后再被进一步判定为某种具体的数值类型. 基本数类型将在 § 2.2 进一步讨论.

```
>type([1, 2, 3]);
List
>type(x^2+3*x+2);
Expression
>type(a);
Variable(UNDEFINED)
```

这里的 a 是未赋值 (UNDEFINED) 的变量 (Variable).

```
>a:=x^2+3*x+2;
x^2+3*x+2
>type(a);
Expression
```

这时 a 已经用一个表达式赋值, 如果再用 type 查询 a, 其数据类型变成 Expression.

有关变量的类型和赋值问题, 在 § 2.3 中有详细讨论.

在 MMP 中, 输入对象的数据类型决定着它的内部数据结构, 同时也决定着在这个对象上可以进行的运算. 用户输入任何一个运算指令后, MMP 会自动对输入对象进行类型检查, 如果发现类型错误就会产生相应的出错信息.

```
>ifactor(123456789.);
Syntax Error!
The argument is not an integer type in function ifactor()!
```

在 MMP 中, 因式分解函数 ifactor 只能作用在整数类型上, 而这里输入的数据是浮点数类型, 因此, MMP 在做类型检查时会发现类型错误并产生出错提示信息. 在此例中若把小数点去掉，则可以得到正确的分解.

```
>ifactor(123456789);
[3,2,3607,1,3803,1]
```

表 2.1 MMP 的数据类型

数据类型	简 称	说 明	举 例
Number	Num	基本数类型	泛指以下任何数的类型
BigNumber	BN	整数类型	1, 20, −100 等
RatNumber	FracN	分数类型	1/2, −5/4 等
FloNumber	FN	浮点数类型 (多精度)	3.14, −0.789, 123. 等
ComNumber	CN	复数类型	2+3*I, 2+3I, 2.5−(2/3I) 等
DouNumber	DN	浮点数类型 (双精度)	3.14D, −0.789D, 123.D 等
Variable	V	变量类型	a, var, x, y 等
Expression	E	表达式类型	$(x+1)\wedge 2, x/y$ 等
Polynomial	P	多项式类型	expand($(x+1)\wedge 2$) 等
Rational	RP	分式类型	expand(x/y) 等
List	[L]	链表类型	[1, 2, 3],[ ] 等
Matrix	[M]	矩阵类型	matrix(2, 2, [1, 2, 3, 4]) 等

MMP 支持的数据类型可分为基本数据类型和复合数据类型两大类. 基本数据类型包括各种数值类型 (整数, 分数, 浮点数和复数), 以及变量, 表达式, 多项式和分式等; 复合数据类型包括链表和矩阵等. 它们都是符号计算中不可缺少的数据类型. 表 2.1 中列出了 MMP 常用的几种数据类型, 有关它们的详细介绍将是本章后面几节的主要内容.

注意, 表 2.1 中的第 1 列给出的是各种数据类型在 MMP 中内设的类型名, 也就是用 type 指令查询得到的结果, 第 4 列给出的是相应数据类型在 MMP 界面上的输入形式. MMP 除提供了上述各种基本数据类型外, 还提供了字符串、向量等

许多相对不太常用的数据类型, 这里没有全部列出.

2. 各种数据类型之间的运算.　由于 MMP 中存在许多数据类型, 因此就存在着不同类型的对象相互运算的可能性. 一元运算 (如一元的 “−”) 的计算结果类型总与运算对象类型相同. 如果参与二元运算的两个运算对象具有相同类型, 计算的结果通常也是这个类型 (有个别例外). 如果两个运算对象的类型不同, MMP 明确地定义了计算结果的类型. 下面给出几个表格, 详细介绍这方面的情况.

表 2.2 和表 2.3 分别给出了加法和乘法运算对不同运算对象的计算结果类型. 由于表格的对称性, 这里只给出其中的上三角部分. 表 2.4 是关于除法的表格, 由于除法运算不具有可交换性, 该表要复杂一些. 最后的表 2.5 是有关指数运算的表格. 这些表中的星号表示没有这种运算情况.

表 2.2　加法运算的数据类型

<table>
<tr><th colspan="2" rowspan="2">add(+)</th><th colspan="5">Num</th><th rowspan="2">V</th><th rowspan="2">E</th><th rowspan="2">P</th><th rowspan="2">RP</th><th rowspan="2">[L]</th><th rowspan="2">[M]</th></tr>
<tr><th>BN</th><th>FracN</th><th>FN</th><th>DN</th><th>CN</th></tr>
<tr><td rowspan="5">Num</td><td>BN</td><td>BN</td><td>FracN</td><td>FN</td><td>DN</td><td>CN</td><td rowspan="5">E</td><td rowspan="5">E</td><td rowspan="5">P</td><td rowspan="5">RP</td><td rowspan="5">*</td><td rowspan="5">*</td></tr>
<tr><td>FracN</td><td></td><td>FracN</td><td>FN</td><td>DN</td><td>CN</td></tr>
<tr><td>FN</td><td></td><td></td><td>FN</td><td>DN</td><td>CN</td></tr>
<tr><td>DN</td><td></td><td></td><td></td><td>DN</td><td>CN</td></tr>
<tr><td>CN</td><td></td><td></td><td></td><td></td><td>CN</td></tr>
<tr><td colspan="2">V</td><td colspan="5"></td><td>E</td><td>E</td><td>E</td><td>E</td><td>*</td><td>*</td></tr>
<tr><td colspan="2">E</td><td colspan="5"></td><td></td><td>E</td><td>E</td><td>E</td><td>*</td><td>*</td></tr>
<tr><td colspan="2">P</td><td colspan="5"></td><td></td><td></td><td>P</td><td>RP</td><td>*</td><td>*</td></tr>
<tr><td colspan="2">RP</td><td colspan="5"></td><td></td><td></td><td></td><td>RP</td><td>*</td><td>*</td></tr>
<tr><td colspan="2">[L]</td><td colspan="5"></td><td></td><td></td><td></td><td></td><td>[L]</td><td>*</td></tr>
<tr><td colspan="2">[M]</td><td colspan="5"></td><td></td><td></td><td></td><td></td><td></td><td>[M]</td></tr>
</table>

§2.2　数 的 运 算

作为符号计算系统, MMP 具有处理无穷精度数值的能力, 这一点不同于一般的程序设计语言. 如在 C, Fortran 等语言中, 各种数据类型都有明确的长度限制, 而 MMP 则不限制整数的大小及浮点数的有效数字位数. 只要计算机内存允许, 这里的数值计算可以得到任意大小、任意精度的结果. MMP 的数值计算涉及整数、分数、浮点数和复数等数值类型的运算, 以及它们的混合运算. 本节将逐一讨论这些数值类型的特性和功能.

表 2.3　乘法运算的数据类型

<table>
<tr><th colspan="2" rowspan="2">mul(*)</th><th colspan="5">Num</th><th rowspan="2">V</th><th rowspan="2">E</th><th rowspan="2">P</th><th rowspan="2">RP</th><th rowspan="2">[L]</th><th rowspan="2">[M]</th></tr>
<tr><th>BN</th><th>FracN</th><th>FN</th><th>DN</th><th>CN</th></tr>
<tr><td rowspan="5">Num</td><td>BN</td><td>BN</td><td>FracN</td><td>FN</td><td>DN</td><td>CN</td><td rowspan="5">E</td><td rowspan="5">E</td><td rowspan="5">P</td><td rowspan="5">RP</td><td rowspan="5">*</td><td rowspan="5">*</td></tr>
<tr><td>FracN</td><td></td><td>FracN</td><td>FN</td><td>DN</td><td>CN</td></tr>
<tr><td>FN</td><td></td><td></td><td>FN</td><td>DN</td><td>CN</td></tr>
<tr><td>DN</td><td></td><td></td><td></td><td>DN</td><td>CN</td></tr>
<tr><td>CN</td><td></td><td></td><td></td><td></td><td>CN</td></tr>
<tr><td colspan="2">V</td><td colspan="5"></td><td>E</td><td>E</td><td>E</td><td>E</td><td>[L]</td><td>[M]</td></tr>
<tr><td colspan="2">E</td><td colspan="5"></td><td></td><td>E</td><td>E</td><td>E</td><td>[L]</td><td>[M]</td></tr>
<tr><td colspan="2">P</td><td colspan="5"></td><td></td><td></td><td>P</td><td>RP</td><td>[L]</td><td>[M]</td></tr>
<tr><td colspan="2">RP</td><td colspan="5"></td><td></td><td></td><td></td><td>RP</td><td>[L]</td><td>[M]</td></tr>
<tr><td colspan="2">[L]</td><td colspan="5"></td><td></td><td></td><td></td><td></td><td>[L]</td><td>[M]</td></tr>
<tr><td colspan="2">[M]</td><td colspan="5"></td><td></td><td></td><td></td><td></td><td></td><td>[M]</td></tr>
</table>

表 2.4　除法运算的数据类型

<table>
<tr><th colspan="2" rowspan="2">div(/)</th><th colspan="5">Num</th><th rowspan="2">V</th><th rowspan="2">E</th><th rowspan="2">P</th><th rowspan="2">RP</th><th rowspan="2">[L]</th><th rowspan="2">[M]</th></tr>
<tr><th>BN</th><th>FracN</th><th>FN</th><th>DN</th><th>CN</th></tr>
<tr><td rowspan="5">Num</td><td>BN</td><td>FracN</td><td>FracN</td><td>FN</td><td>DN</td><td>CN</td><td rowspan="5">E</td><td rowspan="5">E</td><td rowspan="5">P</td><td rowspan="5">RP</td><td rowspan="5">*</td><td rowspan="5">*</td></tr>
<tr><td>FracN</td><td>FracN</td><td>FracN</td><td>FN</td><td>DN</td><td>CN</td></tr>
<tr><td>FN</td><td>FN</td><td>FN</td><td>FN</td><td>DN</td><td>CN</td></tr>
<tr><td>DN</td><td>DN</td><td>DN</td><td>DN</td><td>DN</td><td>CN</td></tr>
<tr><td>CN</td><td>CN</td><td>CN</td><td>CN</td><td>CN</td><td>CN</td></tr>
<tr><td colspan="2">V</td><td colspan="5">E</td><td>E</td><td>E</td><td>E</td><td>E</td><td>*</td><td>*</td></tr>
<tr><td colspan="2">E</td><td colspan="5">E</td><td>E</td><td>E</td><td>E</td><td>E</td><td>*</td><td>*</td></tr>
<tr><td colspan="2">P</td><td colspan="5">P</td><td>E</td><td>E</td><td>RP</td><td>RP</td><td>*</td><td>*</td></tr>
<tr><td colspan="2">RP</td><td colspan="5">RP</td><td>RP</td><td>RP</td><td>RP</td><td>RP</td><td>*</td><td>*</td></tr>
<tr><td colspan="2">[L]</td><td colspan="5">*</td><td>*</td><td>*</td><td>*</td><td>*</td><td>*</td><td>*</td></tr>
<tr><td colspan="2">[M]</td><td colspan="5">*</td><td>*</td><td>*</td><td>*</td><td>*</td><td>*</td><td>*</td></tr>
</table>

1. 整数和分数的运算.　MMP 中的整数和分数的运算都属于精确运算, 其运算结果一定是一个 “精确值”. 例如:

```
>3075/30850;
```

$$\frac{123}{1234}$$

表 2.5 指数运算的数据类型

exp(左 ∧ 上)		Num						V	E	P	RP	[L]	[M]
		BN(+)	BN(−)	FracN	FN	DN	CN						
Num	BN	FracN	FracN	E	E	E	E	E	E	P	RP	*	*
	FracN	FracN	FracN	E	E	E	E						
	FN	FN	FN	E	E	E	E						
	DN	DN	DN	E	E	E	E						
	CN	CN	CN	E	E	E	E						
V		E	E	E				E	E	E	E	*	*
E		E	E	E				E	E	E	E	*	*
P		P	RP	E				E	E	E	E	*	*
RP		RP	RP	E				E	E	E	E	*	*
[L]		E	E	*				*	*	*	*	*	*
[M]		E	E	*				[M]	*	*	*	*	*

MMP 中用 $p/q(p,\ q$ 为整数) 的形式表示分数. MMP 对于这样的算式只作化简操作, 即消去其分子和分母的最大公因数, 最后得到的仍是分数, 而不是某个浮点数 (近似值).

```
>54*2342-57/111+154*25/4000;
```

$$\frac{374346609}{2960}$$

这一示例表明, 整数和分数进行混合运算时, MMP 会得到分数的结果.

```
>n:=123456789^15;
235898216559148381209470363691472039483181699385194041759684258
23418008249115809912616071588527110255905622789563711716349
>num_len(n);
122
```

这里演示的是对一个整数的求幂运算. 可以看到, 即使运算结果是一个很大的整数, MMP 还是能精确地把它计算出来. 函数 num_len 得到的是这个整数的十进制位数, 通过这个函数用户可以直观地感受到运算结果的 "规模".

虽然 MMP 可以表达的整数范围非常大, 但它也有一个上限, 这个上限实际上是由 MMP 中整数存储的数据结构所决定的. 在 MMP 中, 整数 N 被表示为 $N = a_0 + a_1B + a_2B^2 + \cdots + a_{n-1}B^{n-1}$, 其中的 B 称为整数的 "基数", $0 \leqslant a_i < B(i = 0, \cdots, n-1)$ 称为整数中的一个 "字", 而其中 "字" 的个数 n 称为整数 N 的 "长度". MMP 用一个长度为 n 的连续存储块来存放整数 N 的 n 个 "字". 通过各方面因素

的综合考虑, 在 MMP 中所用的 “基数”$B = 2^{32}$, 并且采用 24 个二进制位表示整数存储块的 “长度”. 根据这种设计, 我们不难计算出 MMP 所能表示的最大整数的二进制位数是 2^{29}, 这是一个大约为一亿个十进制位的数, 表示这个数需要 64M 字节内存空间. 这样大的数已足以应付我们在实际应用中可能遇到的整数运算规模.

除了直接用基本算术运算符表示的各种运算外, MMP 还提供了许多整数运算函数. 限于篇幅, 这里不准备逐一介绍这些函数. 有兴趣的读者可以参阅《MMP 用户手册》, 那里给出了 MMP 中所有整数运算的完整介绍. 下面我们将介绍其中一些整数运算, MMP 丰富的整数运算功能可以从中窥见一斑.

```
>iquo(12, 8);
1
>irem(12, 8);
4
>irem(-12, 8);
-4
```

以上是整数的除法运算指令. 前面讲过, “/” 运算符只作化简运算, 而 iquo 和 irem 才作真正的整数除法, 它们分别计算出整数除法的商和余数. 对于两个整数 n 和 m, 显然有 $n = \mathrm{iquo}(n, m) * m + \mathrm{irem}(n, m)$. 需要注意的是, 余数的符号总是与被除数保持一致. 例如：$\mathrm{irem}(-12, -8) = -4$, $\mathrm{irem}(12, -8) = 4$.

除了基本四则运算之外, MMP 还提供了许多与整数有关的常用数学函数, 如 isqrt(开方函数), ilog2(以 2 为底的对数), ilog10(以 10 为底的对数) 等, 它们使用户可以方便地处理各种数学问题. 需要注意, 这些函数返回的结果是不超过实际数值的最大整数, 但不一定是最接近实际数值的那个整数, 例如：

```
>isqrt(15);
3
>ilog2(10);
3
>ilog10(2000);
3
```

稍后我们还将介绍另外两个计算平方根的函数 nsqrt 和 sqrt, 读者应注意它们和 isqrt 的区别.

MMP 中提供了许多关于素数的运算. 考虑第 6 个 Fermat 数：

```
>F:=2^(2^6)+1;
18446744073709551617
>isprime(F);
```

```
0
>ifactor(F)
[274177, 1, 67280421310721, 1]
>nextprime(F);
18446744073709551629
```

函数 isprime 是一个返回值为 1 或 0 的函数, 它判断一个整数是否为素数, 是则返回 1, 否则返回 0. 本例首先判断整数 F 是否为素数, 其输出结果表明它不是素数. 随后函数 ifactor 对整数 F 进行因子分解, 返回由该数的所有质因子及其相应的指数所构成的列表. 这里给出整数 F 的分解结果为

$$2^{2^6} + 1 = 274177 * 67280421310721$$

最后函数 nextprime 给出比 F 大的下一个素数. 再看一个例子:

```
>N:=7818684719769;
>isprime(N);
0
>ifactor(N);
[3,5,7,3,11,2,23,1,37,1,911,1]
>nextprime(N);
7818684719771
```

本例表明整数 N 的分解结果为 $7818684719769 = (3)^5(7)^3(11)^2(23)(37)(911)$.

除了一般意义下的整数运算之外, 有限域上的整数运算也是十分重要的, 符号计算领域的许多算法中都大量用到模 m(m 是个整数, 通常是一个素数) 意义下的整数运算. 鉴于这种情况, MMP 提供了大量与之相关的运算功能.

```
>imod(15, 4);
3
>imod0(15, 4);
-1
>imod_inv(10, 7);
5
>imod_sqrt(100, 7);
4
```

函数 imod 和 imod0 均为取模运算函数, 实际上可以认为它们和 irem 函数类似, 都是用来计算整数除法的余数. imod 以非负的方式求余数, 它所求出的余数均为非负值; imod0 则以对称的方式来求余数, 其余数落在以 0 为中心的某个对称区间内.

imod_inv 计算一个整数在指定有限域上的逆元素. 上例表明, 10 在模 7 的有限域上的逆元素是 5. imod_sqrt 则返回一个整数在指定有限域上的平方根之一. 本例中, 100 在模 7 的有限域上的平方根之一是 4.

2. 浮点数和计算精度. 除了能支持整数和分数运算的精确运算外, MMP 还支持浮点运算 (或称为近似运算). 虽然精确运算可以完全避免算术运算的舍入误差, 保证计算的准确度, 但一般说精确运算的速度比较慢, 占用内存比较多. 在实际应用中, 特别是在工程计算中, 浮点运算也是十分有用的. 它不仅可以使我们对计算结果有更直观的 "量" 的感觉, 而且可以有效地提高计算效率.

MMP 中提供了两种类型的浮点数, 即双精度浮点数和多精度浮点数. 双精度浮点数用于进行双精度的浮点运算, 实际上使用的就是 C/C++ 语言中的 double 类型. 由于在大量的具体应用中, 双精度运算已足够满足实际问题的精度要求, 为了充分利用程序设计语言已有的数据资源, 并且方便统一的处理和管理, MMP 的数值类型中加入了对 double 类型进行封装之后形成的双精度浮点类型, 在用户界面上可以识别. 鉴于双精度浮点数已为人们所熟知, 本节将重点介绍多精度浮点数及其计算精度问题.

MMP 环境中定义了一个特殊的全局变量 Digits, 用于控制浮点运算输出结果的有效数字部分的位数. 目前的默认取值是 Digits=30, 也就是说, 把浮点运算结果设定为保留 30 位有效数字, 这对于大多数的计算应该是足够了. 如果实际应用中有更高精度要求的运算, 30 位有效数字仍然不够, 用户只需修改 Digits 的值, 就可以增加计算的精度 (有效数字的位数). 例如：在默认情况下 Digits=30, 此时有

```
>100./17.;
5.88235294117647058823529411765
```

下面修改 Digits 的值为 50, 同样的运算可以得到更精确的运算结果：

```
>100./17.;
5.8823529411764705882352941176470588235294117647059
```

若把浮点数 (近似值)100. 和 17. 分别改成整数 (精确值)100 和 17, 则有

```
>100/17;
```

$$\frac{100}{17}$$

如前面介绍的, 当 "/" 运算符作用于整数时, 只做化简, 得到分数的结果; 只有当 "/" 运算符作用于浮点数时, 才会去做浮点数的除法, 即进行近似计算, 得到浮点数的结果, 并根据 Digits 所设定的值来确定计算的精度. 注意, 这里运算结果的最后一位数字已考虑到后面的进位.

MMP 的用户还可以用 SetPrecision 指令指定某个浮点运算的精度. 请看下面

的例子：

```
>SetPrecision(100./17., 5);
5.8824
>SetPrecision(100./17., 10);
5.882352941
```

需要注意的是, SetPrecision 指令只是指定某个特定的浮点运算的计算精度, 而全局变量 Digits 则用于控制所有浮点运算的精度. 另外, SetPrecision 指令还可以用来修改浮点数的有效数字位数, 例如：

```
>SetPrecision(1.2345678958476, 10);
1.234567896
```

如果运算表达式中的任何一个运算数是浮点数 (近似数), MMP 就会把遇到的精确数 (如果有) 转换成浮点数, 然后进行近似运算, 最后得到浮点数的结果.

```
>54*2342-57/11+154*25/4000;
```

$$\frac{374346609}{2960}$$

```
>54.*2342-57/111+154*25/4000;
```

126468.448986486486486486486486

MMP 还提供了两个用于取浮点数的整数部分的函数 ceil 和 floor.

```
>ceil(3.5);
4
>ceil(-3.5)
-3
>floor(3.5);
3
>floor(-3.5);
-4
```

ceil 返回不小于参数的最小整数, floor 则返回不大于参数的最大整数. ceil 和 floor 同样可以作用于分数, 功能与此类似.

3. 复数的运算. MMP 中除了可以进行实数的运算, 还可以进行复数运算. 复数的实部和虚部可以是整数、分数和浮点数中的任意一种类型, 但不能再是复数类型. 虚数单位 i(即 $\sqrt{-1}$) 用大写字母 I 表示. 在包含有复数的运算表达式中, 其他数值类型将自动转化成复数类型. 所有的算术运算符对复数类型都适用.

```
>(2+3I)*(5+6I)-(7+4*I)/(1+2*I)+5;
-6+29I
```

这是一个包含复数的四则运算. 表达式中的 5 将自动转化成复数类型参加运算. 注意, 在 MMP 中, 复数在输入时可以写成 a+b*I 的形式. 如果这里的 a 和 b 都是数, 也可以写成 a+bI 的形式, 中间的乘号 * 可以不写.

```
>conjugate (2+3I);
2-3I
>Re(2+3I);
2
>Im(2+3I);
3
```

上面例子显示了求复数的实部, 虚部和共轭复数的三个函数.

```
>abs(2+3I);
3
>abs(2+3.0I);
3.60555127546398929311922126747
>abs(2/3+(3/5)I);
13/15
```

在 MMP 中, 如果绝对值函数 abs 的参数是复数, 算出的是该复数的模. 需要注意, 此时 abs 返回值的类型取决于复数的实部和虚部的类型. 当实部和虚部是同一种类型时, 返回值类型与它们一致; 当实部和虚部不是同一种类型时, abs 的返回值是二者中优先级较高的类型. 有关数值类型的优先级问题在下一小节介绍.

4. 基本数类型和混合运算. 如前所述, MMP 可以表示和处理各种不同类型的数, 包括整数, 分数, 浮点数和复数等. 在 MMP 的用户界面上, 这些不同的数值类型都首先被统一识别为一种抽象的基本数类型. 对于基本数类型, MMP 提供了能够适用于任何一种数值类型的运算和操作, 例如四则运算符, 比较运算符、绝对值函数、平方根函数、符号函数等.

```
>nsqrt(19469237469234124);
139532209
>nsqrt(19469237469234124. );
139532209.432926718778509613389
>nsqrt(19461412341/1234123412);
69752/17565
```

这个例子说明：无论参数是整数、浮点数还是分数, 在 MMP 中都首先被看作是同一种基本数类型. 由于开平方运算可以应用于各种数值类型, 因此 MMP 为基本数

类型提供了开平方函数 nsqrt, 这个函数可以用在任何数值类型上, 其返回值的类型总是与输入参数的具体类型一致. 当函数 nsqrt 用在整数上时, 就相当于前面提过的函数 isqrt. 当然, isqrt 只能用于整数.

我们在 § 2.1 讨论过不同数据类型之间的混合运算问题, 下面我们具体关注各种数之间的混合运算. 看下面的例子：

```
>1234*2333.32-57/11;
2879311.69818181818181818181818
>(1+2I)*12345.678+250*11/17;
12507.4427058823529411764705882+24691.356I
```

如果遇到二元运算符的两个运算对象属于不同的数值类型, MMP 首先自动识别各个运算数的类型, 然后做必要的类型转化, 得到相同类型的数值后再调用具体类型的相应运算. 类型转化将遵循一定的优先级顺序进行. 不同数值类型先转化为其中优先级最高的类型, 然后再针对这种类型进行运算. 目前 MMP 所采用的优先级顺序按从高到低排列依次是

$$复数 > 浮点数 > 分数 > 整数$$

在上例中, 第一个算式中浮点数的优先级最高, 因此最终结果是浮点数类型. 第二个算式里出现了优先级最高的复数类型, 因此最终运算结果是复数. 如前所述, 本例中涉及的浮点数运算都按预设的 30 位有效数字的计算精度进行.

§2.3　变量和赋值语句

在介绍 MMP 的变量之前, 首先需要了解 MMP 的赋值语句. 在 MMP 中, 形如 lhs:=rhs 的语句是赋值语句, 表示将右边式子 rhs(right hand side) 的值赋给左边的式子 lhs(left hand side). 位于赋值符号左边的式子必须是一个变量, 右边的式子可以是任何数学表达式. 赋值语句建立起其左边的变量和一个值的关联. 变量一旦被赋值, 在计算出现这个变量的任何表达式时, 其中的这个变量都将被与之相关联的 "值" 所取代. 这是 MMP 作为一个符号计算语言所特有的赋值机制, 在运用 MMP 时需要特别注意.

```
>x;
x
>x:=y+1:
>x;
y+1
>(x+1)^2;
```

```
(y+2)^2
```

在执行第一条指令时变量 x 还未被赋值, 此时输出 x 时仍旧显示变量名 “x”. 第二条指令是一个赋值语句, 它把表达式 y+1 赋值给变量 x, 此时输出 x 时则显示它所关联的 “值”y+1. 第四条指令是一个表达式, 该表达式中出现了已赋值的变量 x, 输出时 MMP 把 x 替换成了它的 “值”y+1.

MMP 的变量名是可以包含英文字母、数字或下划线的字符串, 其中第一个字符不能是数字. 需要注意的是, MMP 中的保留字, 例如 if、while、do、from 等, 不能被用作变量名. MMP 的系统函数名, 例如：sqrt、abs、length 等, 也不能被用作变量名.

事实上, MMP 变量有许多不同情况, 除了上述通常意义下的变量之外, 还包括一些比较特殊的数据对象, 它们在 MMP 中也被定义为 “变量”. MMP 中的每个变量都带有一个类型标识 (type), 用来描述该变量的具体类型. MMP 的变量有许多类型, 其中用户经常用到的一般只有两类, 分别为未定义型 (UNDEFINED) 和函数型 (TFUNCTION). 下面仅对这两种类型的变量加以说明.

UNDEFINED 指未赋值的符号变量, 例如前面例子中未赋值时的变量 x、y 等. MMP 把一些特殊的数学函数, 例如三角函数 sin(x)、cos(x) 等, 开方函数 sqrt(x)、curt(x) 等也看成是一类特殊的变量, 这就是 TFUNCTION 类型的变量, 也称为特殊函数变量. 用类型查询指令 type 查询变量, 就可以得到有关该变量具体类型的信息. 看下面的例子：

```
>type(x);
```

Variable(UNDEFINED)

```
>x:=2;
2
>type(x);
```

Number(BigNumber)

```
>x:=y^2+2*y+1;
```

y^2+2y+1

```
>type(x);
```

Expression

```
>type(sin(x));
```

Variable(TFUNCTION)

```
>type(sin(x+y));
```

Variable(TFUNCTION)

x 在赋值之前为 UNDEFINED 类型的变量, 赋值之后的数据类型则取决于它所关联的 “值”. sin(x) 和 sin(x+y) 都是 TFUNCTION 类型的变量, 而且被看作是两个不同的变量.

对于 TFUNCTION 类型的变量, 通常存在一个化简的问题, 例如三角函数和开方函数的化简等. 这些操作在 MMP 中都被理解为对相应 TFUNCTION 型变量的化简. MMP 通过 expand 指令完成这类化简运算. 例如：

```
>expand(sin(-x));
```
$-sin(x)$
```
>expand(cos(-x));
```
$cos(x)$
```
>sqrt(18);
```
$\sqrt{18}$
```
>expand(sqrt(18))
```
$3\sqrt{2}$

上例中, 前两条指令进行三角函数的化简运算. 在第三条指令中, sqrt(18) 作为一个 TFUNCTION 型变量, 未进行 expand(展开) 操作时不做任何化简, 第四条指令运用 expand 指令得到化简之后的结果. 与前面曾经提到过的两个开平方函数 isqrt 和 nsqrt 不同的是, 函数 sqrt 通过化简可以得到平方根的精确运算结果.

从后面的章节中读者将会了解到, MMP 里的许多函数运算指令都与变量序有关. 变量序可以通过两种方式确定, 一种是采用系统默认的变量序, 另一种是由用户指定变量序. 在 MMP 启动之后, 系统内部将建立一个初始的系统变量表, 对于所有与变量序有关的指令, 除非特别指定, MMP 都将以系统变量表所给定的变量序作为默认的变量序. 用户也可以通过在函数参数中加入变量列表的方式指定相应运算的变量序. 如不加特殊说明, 变量在变量列表中总按从高到低的次序排列. 在后面的章节中可以看到大量涉及变量序的函数运算指令的例子, 这里不再单独举例.

§2.4 表 达 式

1. 表达式. 在 MMP 中任何由操作数 (operand) 与运算符 (operotor) 所组成的数学式都是表达式. 操作数可以是数和变量 (包括一般符号变量以及特殊函数变量), 而运算符可以包括算术运算符 $(+,-,*,/,\ \wedge$(幂)), 逻辑运算符($\sim$ (not), &&(and), $\|(or)$), 以及关系运算符 $(>, >=, <, <=, ==, !\ =)$, 例如：$3x^2+5x+7$, $\sin(x)+2\cos^2(x)+x$, $\mathrm{sqrt}(x)/(y+1)$ 等都是表达式.

虽然数学里的多项式可以看成是一类特殊的表达式, 但是 MMP 里的多项式有

特殊规定的处理方式. 在 MMP 中, 一个符合多项式定义的表达式只有在执行了 expand 指令之后才能成为多项式类型. 例如：

```
>type(x^2+2*x+1);
Expression
>type(expand(x^2+2*x+1));
Polynomial
>type(x^y);
Expression
>type(expand(x^y));
Expression
>type(sin(x))^2+3*sqrt(2)+cos(x);
Expression
>type(expand(sin(x)^2+3*sqrt(2)+cos(x)));
Polynomial
```

并不是所有的表达式都能展开成为多项式. 如果该表达式不符合多项式的定义, 经过展开操作之后仍是表达式. 例子中的表达式 x∧y 展开之后仍是表达式. 特别需要指出的是, 由于 MMP 把 sin(x), cos(x), sqrt(2) 等数学函数也看作是特殊的变量, 所以像 sin(x)∧2+3*sqrt(2)+cos(x) 一类的含有三角函数与开方函数的表达式, 通过 expand 指令展开后也成为多项式类型.

对于表达式, 系统只会进行一些简单的计算, 例如简单的算术运算, 合并同类项和约分等, 并不进行复杂的计算.

```
>x^2+2*x*y+5*x^2-x*y+5+x^4/x;
```

$6x^2+xy+x^3+5$

```
>(x^3*y)/(x^2*y);
```

x

```
>(x+1)^2-(x+1)*(x+1);
```

0

```
>y*(x*y);
```

xy^2

```
>sin(x)^4/ sin (x)+6;
```

$sin^3(x)+6$

```
>(x+1)^2-x^2-2*x-1;
```

$(x+1)^2-x^2-2x-1$

```
>(x^2 -1)/( x +1);
```

$$\frac{x^2-1}{x+1}$$

2. 表达式的展开和化简. 在上面的例子中, 对最后的两个表达式, MMP 没有做任何化简, 这是因为 MMP 对表达式不自动进行因式分解与展开运算. 如果希望作进一步的计算, 就需要使用 expand 指令. 事实上, expand 指令在 MMP 中兼具两种功能, 即展开的功能和化简的功能. 先看 expand 的展开功能.

```
>expand((x+y)^7);
```

$x^7+7yx^6+21y^2x^5+35y^3x^4+35y^4x^3+21y^5x^2+7y^6x+y^7$

```
>expand((x+y)^2*(x^2+3*x*y+5));
```

$x^4+5yx^3+7y^2x^2+5x^2+3y^3x+10yx+5y^2$

```
>expand((sin(x)+cos(y))^2);
```

$sin^2(x)+2*cos(y)*sin(x)+cos^2(y)$

从以上例子不难看出, expand 指令使符合多项式定义的表达式被当作多项式处理, 从而具备了多项式的功能与操作. MMP 为多项式提供的运算远比为表达式提供的运算丰富得多, 许多在表达式层面上无法进行的运算, 都可以通过 expand 指令得以实现. 例如前面看过的有关表达式计算的两个例子, 下面我们用 expand 指令进行计算:

```
>expand((x+1)^2-x^2-2*x-1);
0
>expand((x^2-1)/(x+1));
x-1
```

以上操作也可以看作是通过 expand 指令对表达式进行化简. 在 § 2.3 中已给出了几个用 expand 对三角函数, 开方函数等数学函数进行化简的例子, 下面再看几个这样的例子.

```
>expand(sqrt(-18));
```

$3I\sqrt{2}$

```
>expand(sqrt(-1/18));
```

$(1/6I)\sqrt{2}$

```
>expand(sqrt(12345^13));
```

$353953788908662482314062 5\sqrt{12345}$

```
>expand(curt(-1/18));
```

$(-1/6)\sqrt[3]{12}$

```
>expand(sqrt(x^2+4*x+4));
```

$x+2$

```
>expand(sqrt(18)^2+2*sqrt(18)+1);
```

$19+6\sqrt{2}$

```
>expand(sin(-x)^2 +2*cos(-x)*sin(-x)+6);
```

$sin^2(x)-2cos(x)sin(x)+6$

这里的前四条指令分别是对具体数值的开平方和开立方运算, 这些运算都得到了精确的结果. 第五条指令说明, 在用 expand 对一个符号表达式的开方运算进行化简时, 并不考虑根号里变量的取值范围, 即进行的是纯粹的符号运算. 最后两条指令表明, expand 可以对较复杂的数学公式进行化简.

§2.5 多项式和分式

MMP 的多项式是由变量和算术运算符 $+$, $-$, $*$, $\wedge$ 所组成的一类特殊表达式, 其指数必须是非负整数, 其系数可以是包括整数、分数、浮点数和复数在内的任意数值类型. 其中的变量可以是一般符号变量, 也可以是特殊函数变量.

1. 多项式的排序问题. 这里的排序是指多项式中的项的次序. 与其他符号计算软件不同, MMP 对多项式的排序是自动完成的, 也就是说, 当我们输入一个多项式, 按 Enter 键时, MMP 将自动对该多项式进行重新排序. 看下面的例子：

```
>P1:= x^4+1+x^2+x+x^8 +x^3+x^5;
```

$x^8+x^5+x^4+x^3+x^2+x+1$

```
>P2:=x^4-x^2*y^2+x^3*y+x*y^3-2*y^4;
```

$x^4+x^3y-x^2y^2+xy^3-2y^4$

可以看出, MMP 采用的是纯字典序降幂排列. 变元序可以通过在函数参数中加入变元表的方式加以指定. 若无特殊说明, 通常假定变元表中先出现的变元次序高, 即若给定变元表为 $[x_1,x_2,x_3]$, 则有 $x_1>x_2>x_3$ (这里用大于符号形象地表示变元序的高于关系). 另外, MMP 还提供了一个指令 collect, 借助它可以把多项式按某个指定变量的降幂排列, 或者说按某个指定变量合并同类项. 例如：

```
>p:=5*x^5*y^3+x^5*y^2+x^4*y^2+2*x^3*y+x^5+x^4+x^3+5*x^2+10;
>collect(p, x);
```

$(5y^3+y^2+1)x^5+(y^2+1)x^4+(2y+1)x^3+(5)x^2+(10)$

```
>collect(p, y)
```

$(5x^5)y^3+(x^5+x^4)y^2+(2x^3)y+(x^5+x^4+x^3+5x^2+10)$

在上面例子里, 可以看到在多项式被按某个变量的降幂排列的同时, 其系数多项式也同样按纯字典序的降幂排列.

2. 多项式的基本信息. 获取多项式的各项基本信息, 如系数、次数和项数等, 是多项式运算及编程中非常有用的基本操作, MMP 提供了一系列相关指令. 本节将选

择其中一些比较常用的操作进行简单介绍. 有关多项式的大多数基本操作指令都需要指定变元或者变元序, 若未加指定, 则表明该函数指令是针对默认主变元或者默认变元序进行的. 有关默认变元序的内容已在 § 2.3 中讲过, 这里不再重复.

函数 degree 和 ldegree 分别给出多项式关于指定变元的最高和最低次数. 如果没有指定变元, degree 和 ldegree 将分别返回多项式 p 各项全次数的最高次数和最低次数.

```
>p:= x^3*y^3+x*y^2+y^4:
>degree(p, x);
3
>ldegree(p, x);
0
>degree(p);
6
>ldegree(p);
3
```

前面曾经说过, 表达式只有在执行了 expand 指令之后才能转化为多项式类型, 执行多项式的操作. 事实上, 对于处理多项式的大部分函数指令来说, 并不需要在使用它们之前先去展开输入的表达式, 因为 MMP(对于多项式函数) 会自动完成这一操作, 从而确保得到正确的结果：

```
>degree(x^2-x*(x -1), x);
1
```

函数 coeff 和 coeffs 分别提取多项式某个特定项的系数和多项式的所有系数. 请看下面的例子：

```
>p:=3*x^2+5*y*x+2/3*z*x+2*y^2+2/3*z*y+20;
```

$3x^2 + 5yx + 2/3zx + 2y^2 + 2/3zy + 20$

```
>coeff(p, x, 2);
3
>coeff(p, x, 1);
5y+2/3z
>coeff(p, x, 0);
```

$2y^2 + 2/3zy + 20$

```
>coeffs(p, x);
```

$[3, 5y + 2/3z, 2y^2 + 2/3zy + 20]$

```
>coeffs(p);
```

```
[3, 5, 2, 2/3, 20]
```

函数 coeff 求出多项式关于某个指定变元的指定幂次的系数, 函数 coeffs 给出多项式的所有系数. 需要注意的是, 如果只给 coeffs 一个参数 (一个多项式), 返回的是该多项式关于所有变元的系数表, 即所谓基本系数的系数表, 这是一个数的链表, 表中不包含重复的系数; 若给 coeffs 提供两个参数, 函数返回多项式关于该指定变元的系数表, 这是一个多项式的链表, 表中各系数多项式按指定变元的次数从高到低排列.

如果希望得到多项式关于其主变元的最高次项的系数 (即首项系数或初式), 那么可以调用函数 lead_coeff . lead_coeff 中的第二个参数是一个变元表, 用来指定多项式的主变元. 下面是一个求首项系数 (初式) 的例子:

```
>p:=(3*y^2+2*y+1)* x^3+2*y*x^2+4*x +6;
>lead_coeff(p, [x, y]);
```

$3y^2+2y+1$

```
>lead_coeff(p, [y, x]);
```

$3x^3$

上面第二条指令中指定 x 为主变元, 第三条指令中指定 y 为主变元. 还可以用函数 lead_term 和 reductum 分别求得多项式 p 的首项和余项:

```
>lead_term(p, [x, y]);
```

$3x^3y^2$

```
>reductum(p, [x, y]);
```

$2x^3y+2x^2y+x^3+4x+6$

和多项式的次数一样, 多项式的项数也是反映其规模大小的一个重要的基本信息, 我们可以用 MMP 所提供的 plength 函数获得多项式的项数.

```
>p:=( x^2+2*x +1 )*y^3 +2*x*y^2 +4*y+6;
>plength(p);
6
>plength (p, x);
3
```

从以上例子不难看出, 如果不指定变元, 则 plength 返回多项式完全展开以后的项数, 否则就返回多项式关于某个指定变元的项数.

3. 多项式的基本运算.　多项式的基本运算包括四则运算 “+, −, ∗, /” 和乘方运算 “∧”, 还有一些基本的算术运算函数. 下面是一些简单的例子:

```
>P1:=expand(x^3*y+x+1);
>P2:=expand(x^2+x*y+1);
```

```
>P1+P2 ;
```
$2+x^3y+x^2+xy+x$

```
>P1-P2;
```
x^3y-x^2-xy+x

```
>P1*P2;
```
$yx^5+y^2x^4+yx^3+x^3+yx^2+x^2+yx+x+1$

```
>P1/P2;
```
$\dfrac{1+x^3y+x}{1+x^2+xy}$

```
>expand(P1^3);
```
$y^3x^9+3y^2x^7+3y^2x^6+3yx^5+6yx^4+3yx^3+x^3+3x^2+3x+1$

请注意，如果不作展开运算, 多项式会被当成表达式.

```
>(x^2 -1)/(x+1);
```
$\dfrac{x^2-1}{x+1}$

```
>expand ((x^2 -1)/( x +1));
```
$x-1$

如果需要做真正的除法运算, 那么就应该使用运算函数 quo 和 rem. 在以下例子中, 假定 P1 和 P2 仍然沿用上面例子中的定义.

```
>quo(P1, P2);
```
$yx-y^2$

```
>rem(P1, P2);
```
$y^3x-yx+x+y^2+1$

```
>quo(P1, P2, x);
```
$yx-y^2$

```
>rem(P1, P2, x);
```
$y^3x-yx+x+y^2+1$

quo 和 rem 分别求出两个多项式相除的商 (quotient) 和余式 (remainder). 对于两个多项式 $p1$ 和 $p2$, 总有

$$p1=\text{quo}(p1,p2)*p2+\text{rem}(p1,p2)$$

对于函数 quo 和 rem, 如果在函数调用中没有指定变元, 那就表示关于除式的默认主变元作除法；如果指定了变元, 那么就表示关于该变元作除法. 另外, MMP 还提供了函数 divide, 其功能是测试一个多项式是否被另一个多项式整除, 是则返回 1, 否则返回 0. 例如：

```
>divide(x^3 -1, x-1);
1
```

```
>divide(x^3 -1, x+1);
0
```

多项式的伪除法运算在数学机械化方法中起着非常重要的作用, 它是特征列方法实现的基础. 在 MMP 中, 多项式的伪除法通过函数 pquo 和 prem 来实现. 下面对前面定义的两个多项式 P1 和 P2 作伪除法运算.

```
>pquo(P1, P2, x);
```

$yx - y^2$

```
>prem(P1, P2, x);
```

$y^3x - yx + x + y^2 + 1$

在这个例子中, 多项式 P1 和 P2 关于变元 x 的余式和伪余式是相同的. 我们再看一个余式和伪余式不同的例子. 为此重新定义 P1 和 P2 如下：

```
>P1:=2*x^3*y+x+1;
>P2:=3*x^2+x*y+1;
>rem(P1, P2, x);
```

$1 + x - \frac{2}{3}xy + \frac{2}{9}y^2 + \frac{2}{9}y^3x$

```
>prem(P1, P2, x);
```

$2y^3x - 6yx + 9x + 2y^2 + 9$

4. 多项式的最大公因子和因式分解. 多项式最大公因子 (GCD) 的计算和因式分解是符号计算中的两个重要运算, 在符号计算的许多相关算法中, 都要涉及到多项式最大公因子的计算和因式分解. 因此, 高效快速地实现这两种运算对于整个符号计算系统效率的提高有着十分重要的意义. MMP 中提供了两个函数 gcd 和 factor, 分别用于进行多项式最大公因子的计算和因式分解. 请看例子：

```
>P1 := expand(5*(x+y)*(x-z)^2 ):
>factor(P1);
```

$[[5], 1, [x - z], 2, [x + y], 1]$

```
>P2 := expand((x+y)*( x^3 +5)*( y^2+x*z+1 )^2):
>factor(P2);
```

$[[1], 1, [x^3 + 5], 1, [x + y], 1, [z * x + y^2 + 1], 2]$

```
>gcd(P1, P2);
```

$x + y$

上面例子中首先分别对 P1 和 P2 做因式分解, 可以看到它们都含有因子 $x+y$, 由最后一个计算可知, 这两个多项式的最大公因子就是 $x+y$. 函数 factor 除了可以对一般的多项式进行分解之外, 还可以对一个由特殊函数 (如 $\sin(x)$, $\cos(x)$ 等) 所组成的表达式进行分解, 例如：

```
>factor((sin(x))^2+2*sin (x)*cos (y)+(cos(y))^2);
```
$(sin(x)+cos(y))^2$

需要注意的是, 函数 gcd 和 factor 都是针对整系数多项式的操作, 如果多项式的系数中出现了分数, 那么就很可能得到错误结果. 比如以下计算最大公因子的例子：

```
>gcd((x +(1/2)* y)*( x+z), ( x+(1/2)* y)* x);
1
```

在这种情况下, 我们需要调用另外一个求最大公因子的函数 pgcd, 它适用于任何有理系数多项式的最大公因子计算.

```
>pgcd((x +(1/2)* y)*( x+z), ( x+(1/2)* y)* x);
```
$2x+y$

注意, pgcd 没有返回 $x+\frac{1}{2}y$, 而是返回 $2x+y$, 这是因为我们要求 pgcd 总是返回一个整系数多项式. 对于函数 factor , 它只能将多项式在整数范围内因式分解. 如果用户想进一步在代数数范围内分解, 就要调用函数 factoras. 例如：

```
>factor(x^3+5);
```
$[[1],1,[x^3+5],1]$

```
>factoras(x^3+5, [a^3-5], [a, x]);
```
$[x^2-ax+a^2,1,x+a,1]$

多项式 x^3+5 在整数范围内已无法分解, 而在代数数范围内仍可以分解. 函数 fartoras 实际上是将多项式在代数扩域上进行因式分解. factoras 中的第二个参数应该是一个升列 (本例是由一个多项式所组成的升列), 用于定义因式分解所在的代数扩域, 第三个参数是一个变元表. 注意 factoras 的返回值在形式上与 factor 稍有不同. 在本例中该返回值所代表的含义是 x^3+5 在升列 a^3-5 所定义的代数扩域上的分解结果为 $x^3+5=(x^2-\sqrt[3]{5}x+(\sqrt[3]{5})^2)(x+\sqrt[3]{5})$.

5. 多项式的其他运算函数. MMP 系统为多项式提供了大量的运算函数, 涵盖了几乎所有涉及多项式的功能与操作. 限于篇幅, 这里不可能一一介绍. 本节将介绍几种比较重要而且常用的多项式运算函数.

变量替换是多项式运算里非常重要的一类操作, 对多项式的推导和求值运算都十分有用. 在 MMP 中, 实现多项式变量替换的函数是 subs.

```
>P:= x^2*y^3+x*y+5;
>subs(P, x, 2);
```
$4y^3+2y+5$

```
>subs(P, x, z+1);
```
$z^2y^3+2zy^3+y^3+zy+y+5$

```
>subs(P, [x,y], [1,2]);
```
15
```
>subs(P, [x,y], [x+1,y+1]);
```
$y^3x^2+3y^2x^2+3yx^2+x^2+2y^3x+6y^2x+7yx+3x+y^3+3y^2+4y+7$

可以看出, 函数 subs 把多项式中的指定变元替换为一个数或一个多项式. 该函数既可以支持对多项式中单个变元的替换, 也可以支持多个变元的替换. 实际上, 函数 subs 不仅可以对多项式进行操作, 还可以对任意表达式进行操作, 被替换的对象也可以是任意表达式. 后面还将有单独的一节讨论这个函数. 这里只介绍了它针对多项式的应用.

除了变量替换, 我们还可以对多项式进行一些比较复杂的数学运算, 这些运算函数不仅本身非常重要, 而且在许多涉及多项式的相关算法的实现中也起着十分重要的作用. 下面是两个例子：
```
>P:=x^2*y+2*x*y+3*y^3;
>derivative(P, x);
```
$2xy+2y$
```
>derivative(P, y);
```
$9y^2+x^2+2x$
```
>resultant((x+y)^4, y+z, y);
```
$x^4-4zx^3+6z^2x^2-4z^3x+z^4$

函数 derivative 和 resultant 分别计算出一个多项式的导数多项式和两个多项式的结式.

MMP 还提供了一系列多项式在有限域上的运算指令, 而有限域上的多项式运算通常是多项式的许多相关算法实现的基础. 下面是一个关于多项式的中国剩余定理的例子：
```
>pchrem([3, 7, 11], [2*x^2, 5*x^3, 7*x^6]);
```
$84x^6+33x^3+77x^2$

函数 pchrem 的第一个参数是一个两两互素的整数表, 代表一组模, 第二个参数是一个多项式的链表. 这个函数实现中国剩余算法, 返回一个满足要求的多项式.

6. Gröbner 基. Gröbner 基是关于多项式理想的一个基本算法, 可以用来解决多项式理想的生成元问题.

设 $\mathbb{X}=\{x_1,\cdots,x_n\}$ 是一个变元集合, $\mathbb{K}$ 是一个特征为零的域, $\mathbb{K}[\mathbb{X}]$ 是系数在 $\mathbb{K}$ 中、变量在 $\mathbb{X}$ 中的多元多项式环. $\mathbb{K}[\mathbb{X}]$ 中的一个多项式集合 I 称为一个理想, 如果

- 若 $P\in I, Q\in I$, 则 $P+Q\in I$;

- 若 $P \in I$, 则对于所有 $R \in \mathbb{K}[\mathbb{X}]$，$RP \in I$.

多项式集合 $B = \{P_1, \cdots, P_r\}$ 称为理想 I 的一组基, 如果 I 中的多项式可以写为 B 中多项式的线性组合. 由 B 生成的理想记为 (B). 由著名的 Hilbert 有限基定理, 任意一个多项式理想总有一组有限基. 多项式理想的一个基本问题是所谓理想成员问题, 即给定有限个多项式组成的集合 $\mathbb{P}$ 与一个多项式 P, 判定 P 是否属于由 $\mathbb{P}$ 生成的理想. 这一问题可以由Gröbner 基解答.

我们可以在所有关于变量 $\mathbb{X}$ 的单项式之间建立所谓的字典序. 我们称单项式 M 比 N 的序高, 记做 $M = \prod_{i=1}^{n} x_i^{d_i} > N = \prod_{i=1}^{n} x_i^{e_i}$, 如果存在一个 $1 \leqslant k \leqslant n$, 使得 $d_k > e_k$, 并且对于 $i = k+1, \cdots, n$, 有 $d_i = e_i$. 设 P 为一个多项式, 我们将其中的单项式按照字典序排列：

$$P = c_0 M_0 + c_1 M_1 + \cdots + c_s M_s \tag{2.1}$$

其中 $M_0 > M_1 > \cdots > M_s$. 我们称 M_0 为 P 的首项, c_0 为 P 的首项系数. P 的首项与首项系数分别记为 $\mathrm{lt}(P)$ 与 $\mathrm{lc}(P)$.

一个多项式 Q 称为对于多项式 P 是 *G 约化* 的, 如果 Q 中的单项式都不是 $\mathrm{lt}(P)$ 倍数. 一个多项式 Q 称为对一个多项式集合 $\mathbb{P}$ 是 *G 约化* 的, 如果 Q 对于 $\mathbb{P}$ 中的所有多项式 G 约化.

对于多项式 Q 与 P, 我们定义 Q 关于 P 的 *G 余式* 如下：

1. 如果 Q 对于 P 是 G 约化, 返回 Q;
2. 否则, 设 M 为 Q 中的一个单项式是 $\mathrm{lt}(P)$ 的倍数. 设 $M = N\mathrm{lt}(P)$, 且 c 是 Q 中 M 的系数. 令 $Q = Q - \frac{c}{\mathrm{lc}(P)} \cdot N \cdot P$. 显然 Q 中将不再有 M 项, 而且新增加的项比 M 次序要低. 返回第一步.

记以上 G 余式为 $\mathrm{grem}(Q, P)$. 则有

$$Q = UP + \mathrm{grem}(Q, P)$$

其中 UP 中的单项式的次序不高于 $\mathrm{lt}(Q)$. $\mathrm{grem}(Q, P)$ 对于 P 是 G 约化的.

一个多项式 Q 对一个多项式集合 $\mathbb{P}$ 的 *G 余式* $\mathrm{grem}(Q, \mathbb{P})$ 定义为 Q 关于 $\mathbb{P}$ 中的多项式连续做余式, 直到余式对于 $\mathbb{P}G$ 约化为止. 由于 G 余式的性质, 这一计算过程总是会终止的. 很显然, 如果 $\mathrm{grem}(Q, \mathbb{P}) = 0$, 则 Q 可以写为 $\mathbb{P}$ 中的多项式的线性组合，从而属于由 $\mathbb{P}$ 生成的理想 $(\mathbb{P})$.

一个有限多项式集合 $\mathcal{B}$ 称为Gröbner 基, 如果对于所有的 $P \in (\mathcal{B})$, $\mathrm{grem}(P, \mathcal{B}) = 0$. 显然, 如果 $\mathcal{B}$ 是Gröbner 基, 则可以利用 G 余式来判定任意一个多项式是否属于理想 $(\mathcal{B})$.

Gröbner 基$\mathcal{B}$ 称为 约化 的, 如果 $\mathcal{B}$ 中的多项式 P 对于 $\mathcal{B}-\{P\}$ 约化. 设 $\mathcal{B}$ 是一个Gröbner 基, 则一个等价的 (生成同一个理想) 约化Gröbner 基 可以通过计算 $\mathcal{B}$ 中所有多项式 P 对于 $\mathcal{B}-\{P\}$ 的 G 余式得到. 计算Gröbner 基的算法由 Buchberger 给出 ([9]).

MMP 中的函数 gb 可以用来计算一个多项式集合的Gröbner 基. gb 的调用格式为 gb(ps) 或 gb(ps, 1). 其中参数 ps 是一个多项式集合. 这一命令将用 Buchberger 的算法计算 ps 在字典序下的 Gröbner 基. 这里变量的次序由其出现先后确定：先出现的变量次序高. 如果变量已经在以前出现，则使用以前的顺序. gb(ps) 计算约化的 Gröbner 基, gb(ps,1) 计算非约化的 Gröbner 基. 例如, 所谓 3 循环问题的 Gröbner 基可以如下计算：

```
>gb([x3+x2+x1, x3*x2+x2*x1+x1*x3, x3*x2*x1-1]);
[x3+x2+x1, -x2^2-x1*x2-x1^2, -x1^3+1]
```

这里假定变量顺序为 $x_3 > x_2 > x_1$.

MMP 中还有一个函数 gbs 可以用来计算一个多项式集合的分解Gröbner 基. gbs 的调用格式为 gbs(ps) 或 gbs(ps, 1), 其中参数 ps 是一个多项式集合. 这一命令将计算 ps 在字典序下的分解 Gröbner 基集合. 在计算的过程中, 如果可能, 将对多项式进行因式分解. 举例如下. 如果计算多项式集合 $\mathbb{P}\cup\{UV\}$ 的Gröbner 基, gbs 将分别计算 $\mathbb{P}\cup\{U\}$ 与 $\mathbb{P}\cup\{V\}$ 的Gröbner 基. 这里变量的次序的规定与函数 gb 相同. gbs(ps) 计算约化的 Gröbner 基集合, gbs(ps, 1) 计算非约化的 Gröbner 基集合. 同样考虑 3 循环问题, gbs 将返回三个 Gröbner 基：

```
>gbs([x3+x2+x1, x3*x2+x2*x1+x1*x3, x3*x2*x1-1]);
[[x3+x2+1, x2^2+x2+1, x1-1],
[x3+x1+1, x2-1, x1^2+x1+1],
[x3-1, x2+x1+1, x1^2+x1+1]]
```

由此可以得到方程的六组解：

$$
\begin{aligned}
x_1 &= 1, \quad x_2 = \frac{-1 \pm \sqrt{-3}}{2}, \quad x_3 = -1 - x_2 \\
x_1 &= \frac{-1 \pm \sqrt{-3}}{2}, \quad x_2 = 1, \quad x_3 = -1 - x_1 \\
x_1 &= \frac{-1 \pm \sqrt{-3}}{2}, \quad x_2 = -1 - x_1, \quad x_3 = 1
\end{aligned}
$$

可以看到, 用 gbs 返回的值求解方程更加容易.

7. 分式. 分式又称有理函数, 指形如 $f(x)/g(x)$ 的表达式, 其中 $f(x)$ 与 $g(x)$ 为两个多项式, 而且 $g(x) \neq 0$.

MMP 提供了两个函数 getnu 和 getde, 分别提取出分式的分子与分母.

```
>P:=(x^2+1)/(y-2);
>getnu(P);
```
x^2+1

```
>getde(P);
```
$y-2$

在 MMP 中可以对分式进行四则运算. 下面是几个简单的例子：

```
>expand(x/(y+1)+( x+1)/ y);
```
$$\frac{2yx+x+y+1}{y^2+y}$$

```
>expand((x/(y+1))/(( x+1)/ y));
```
$$\frac{yx}{yx+x+y+1}$$

需要注意的是, 在通常情况下, 要得到分式四则运算的结果, 就必须使用 expand 指令. 若只是直接进行运算, 则 MMP 会原封不动地把原式显示出来.

```
>x^2/(x+1)+(2+x)/(y^2-x);
```
$$\frac{x^2}{x+1}+\frac{2+x}{y^2-x}$$

MMP 是通过 expand 指令实现分式的通分运算的. 同样, 如果需要将分式约分为最简分式, 也需要调用 expand 来完成, MMP 不会自动地去化简分式, 例如：

```
>(x^2-x-6)/(x^2+7*x+10);
```
$$\frac{x^2-x-6}{x^2+7*x+10}$$

```
>expand((x^2-x-6)/(x^2+7*x+10));
```
$$\frac{x-3}{x+5}$$

§2.6 链表的运算

前面几节分别介绍了包括整数、分数、浮点数和复数在内的各种数值类型, 以及变量、表达式、多项式和分式等基本数据类型, 这些类型都属于针对单个数据元素进行操作的简单数据类型. 为了更加灵活地应用 MMP 进行符号运算, 我们还需要用到各种复合数据类型. 复合数据类型是指由一种或几种简单数据元素按照一定规则组合而成的类型, 其意义在于可以把这些数据元素作为整体进行运算或者进行相关的操作. 本节和下一节将介绍 MMP 中的两种最主要的复合数据类型 — 链表和矩阵.

1. 链表. MMP 的链表是用方括号 “[ ]” 括起来的一组以逗号分隔的数据元素的形式表示的, 这些数据元素可以属于任何一种或者几种 MMP 能识别的数据类型.

MMP 中的链表与数学中集合的概念不同. 链表的元素有先后顺序, 而且可以重复. MMP 又为链表提供了类似于数学中集合的基本运算：包括求交集、并集和差集的运算.

现在先来看几个链表的定义：

```
>L1:=[1, 2, 3, 2, 5];
>L2:=[x+y, x*y, x, y, x+y, x^2+2*x*y+1];
>L3:=[20, 5, x^2+1, x/(x+y), 2/3, sin(x+1)+3];
>L4:=[3, 1, [4, 5, [6, 7]], 1, 5];
>L5:=[];
```

上面例子里的 L1 和 L2 分别是整数和多项式的链表, 它们都是由单一类型的数据元素组成的链表, 而 L3 则是一个由多种类型的数据元素组成的链表. MMP 还支持多层嵌套的链表, L4 就是一个嵌套链表, 其中的第 3 个元素为一个子链表, 而这个子链表的第 3 个元素又为一个子链表. L5 是一个空表, 即不包含任何元素的链表.

MMP 提供了一些用于访问链表元素的函数指令. 例如现在需要访问上例链表 L4 中的元素：

```
>part(L4,1);
3
>part(L4,3);
[4, 5, [6, 7]]
>first(L4);
3
>last(L4);
5
```

容易看出, 函数 part 的第一个参数是被访问的链表, 第二个参数是要访问的元素在链表中的索引值（下标）. 注意, 元素下标从 1 开始计数. 函数 first 和 last 分别提取出链表的第一个元素和最后一个元素.

链表的常用操作和运算可以按功能进行分类, 大体分为插入、删除、查找、替换等基本操作, 以及交集、并集、差集等基本运算. 下面将分别介绍其中的一些主要的运算指令.

2. 链表的基本操作. 前面已经介绍了访问链表元素的函数指令. 求链表的长度也是一个非常基本的操作. 有关的函数指令是 nops, 例如：

```
>nops([1, 2, [3, 4]]);
3
```

```
>nops([]);
0
```

请注意, 对于嵌套链表, nops 给出的是它的第一层的元素个数. 空表的长度为 0, 这也可以作为判断一个链表是否为空的依据.

插入和删除是链表的两种最常用操作, MMP 提供了许多进行此类操作的函数指令. 下面介绍一些简单的例子:

```
>insert([1, 2, 3, 4], a, 1);
[a, 1, 2, 3, 4]
>insert([1, 2, 3, 4], a, 2);
[1, a, 2, 3, 4]
>insert([1, 2, 3, 4], a, 5);
[1, 2, 3, 4, a]
>append([1, 2, 3, 4], a);
[1, 2, 3, 4, a]
>append([1, 2, 3, 4], 3);
[1, 2, 3, 4]
>append([a, b, c, d], [e, f]);
[a, b, c, d, [e, f]]
>merge([a, b, c, d], [e, f]);
[a, b, c, d, e, f]
>remove([10, 20, 30, 40], 3);
[10, 20, 40]
>rest([10, 20, 30, 40]);
[20, 30, 40]
```

函数 insert 将元素插入链表的指定位置之前. 例子里的前三条指令就是通过 insert 分别将元素 a 插入原表的表头, 表中和表尾. 函数 append 完成将其第二个参数作为元素插入链表表尾的操作. 需要注意的是, 若被插入的元素已在链表中, append 就直接返回原表. 函数 merge 将两个链表的元素合并 (连接为一个表). 函数 remove 删去链表中的指定元素, 其第二个参数是被删去元素在链表中的索引值（下标）. 函数 rest 得到链表删去第一个元素之后的链表.

3. 链表之间的运算. 前面已经讲过, MMP 中的链表有类似于数学中集合的基本运算, 可以看作集合求交集、并集和差集. 对应的函数指令分别为 intersect、union 和 minus .

```
>union([[1, 2, 3, 4], [3, 2, 5]);
[1, 2, 3, 4, 5]
>union([3, 2, 5], [1, 2, 3, 4]);
[3, 2, 5, 1, 4]
>intersect([1, 2, 3, 4], [3, 2, 5]);
[2, 3]
>intersect([3, 2, 5], [1, 2, 3, 4]);
[3, 2]
>minus([1, 2, 3, 4], [3, 2, 5]);
[1, 4]
>minus([3, 2, 5], [1, 2, 3, 4]);
[5]
```

从以上例子可以看出, 由于链表的有序性, 对于它的交、并、差的运算都不满足交换律, 这是链表的运算与数学中集合的运算的不同之处. 链表运算的一个基本原则是尽量保持原表中元素的顺序关系不变. 以并集运算为例, MMP 首先保留第一个链表的全部元素, 再把第二个链表中的不重复的元素依次加在最后, 从而构造出两个链表的并集.

4. 链表的其他操作. 除上述基本操作与运算之外, MMP 还为链表提供了许多其他的操作. 限于篇幅, 我们只介绍其中的几个比较重要的操作, 例如查找、替换、求逆等. 请看下面的例子:

```
>find(a, [1, 2, a, b, c, a]);
3
>find(a, [1, 2, 3, 4]);
0
```

利用函数 find 可以判断一个指定元素是否在一个链表中, 是则返回它在链表中第一次出现的位置 (索引值), 否则返回 0.

```
>replace([1, 2, 3, 2, 4], a, 2);
[1, a, 3, 2, 4]
>replace([a, b, c, d], [c, d], 2);
[a, [c, d], c, d]
```

函数 replace 把链表中指定位置上的元素替换为新的元素. 注意, 函数的第三个参数是被替换的元素在链表中的索引值 (下标).

```
>inverse([1, 2, 3, 4]);
```

```
[4, 3, 2, 1]
```

函数 inverse 返回指定链表的逆序表.

§2.7 矩阵与线性方程组求解

矩阵是 MMP 中另一种重要的复合数据类型. 本节将详细介绍在 MMP 中如何进行矩阵的基本操作和运算, 以及如何利用矩阵运算求解线性方程组.

1. 矩阵的建立. MMP 可以允许矩阵的元素是任何类型的数据. 下面我们从建立一个矩阵开始, 全面而系统地介绍 MMP 的矩阵实现方案.

矩阵是通过 matrix 指令建立的. 请看下面例子：

```
>A:=matrix(3, 3, [1, 2, 3, 4, 5, 6, 7, 8, 9]);
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```
>type(A);
```

Matrix

```
>B:=matrix(3, 3, [a, b, c, d, e, f, g, h, i]);
```

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

上面示例里用 matrix 函数定义了一个数值矩阵 A 和一个符号矩阵 B. matrix 的前两个参数分别代表矩阵的行数和列数, 第三个参数是个一维链表, 用于给出矩阵的元素. 可以看到, 矩阵在 MMP 中内设的类型名是 Matrix. 需要注意的是, MMP 处理矩阵类型的对象的方式和处理简单数据类型的单个数据元素一样, 采取的是 "自动求值" 方式, 而不是像 Maple 系统所采取的 "按名求值". 所谓 "自动求值", 指系统在显示和计算矩阵时将自动读取矩阵类型的变量名所指向的数据, 而不需要强制调用显示或求值的函数. 上例中, 需要显示矩阵 A 时, 只要键入该变量名即可. 在稍后的 "基本矩阵运算" 中, 还可以进一步了解矩阵在计算中的 "自动求值" 机制.

除了上例中的调用形式外, matrix 函数还有另一种调用形式, 可以更方便地定义只有一行或一列的特殊矩阵：

```
>C1:=matrix([1, 2, 3], 0);
```

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

```
>C2:=matrix([1, 2, 3], 1);
```

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

此时 matrix 函数的第一个参数是用于给出矩阵元素的一维链表, 第二个参数用于定义该矩阵的形状. 取 0 值时表示定义的是 $n \times 1$ 的矩阵, 即列向量; 取非 0 值时表示定义的是 $1 \times n$ 的矩阵, 即行向量. 利用 matrix 的这种调用形式, 可以方便而直接地生成向量. 注意, 在 MMP 中, 向量没有定义为一个独立的数据类型, 而是被看作一种特殊的矩阵.

上面利用函数 matrix 的两种调用形式分别定义了矩阵 A, B, C1 和 C2, 这四个矩阵将在本节后面的示例中经常出现, 届时就不再特别说明了.

2. 矩阵的基本信息. 在利用矩阵运算解决线性代数问题时, 经常需要提取或修改矩阵的基本信息, 例如获取矩阵的维数, 访问和修改矩阵元素, 删除矩阵的行与列, 提取子矩阵等. MMP 提供了许多这方面的函数指令, 以方便用户对矩阵进行各种操作. 下面我们将分别介绍这些指令.

矩阵的行与列的维数可以通过函数 rowdim 和 coldim 获得, 例如：

```
>rowdim (A); coldim (A);
3
3
>rowdim (C1); coldim (C1);
3
1
```

访问和修改矩阵中的元素是矩阵的最常用操作之一, MMP 中相关的函数包括：getval、setval、col、row 和 submatrix, 其中 getval 和 setval 分别用来提取和修改矩阵中的单个元素, 而 col、row 和 submatrix 则可以看作是用来提取矩阵中的多个元素的操作. 下面通过一些例子说明这些函数的用法.

```
>getval(A, 2, 3);
6
>setval(A, 2, 3, 60);
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 60 \\ 7 & 8 & 9 \end{bmatrix}$$

```
>A;
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

函数 getval 用来访问矩阵中的元素. 它的第二、三个参数是该元素在矩阵中的索引值（下标）. 注意, 矩阵元素的两个下标都是从 1 开始计数. 如果要修改矩阵中的元素, 则需要调用函数 setval, 该函数将返回修改之后的矩阵, 原矩阵保持不变. 除了提取矩阵中的单个元素外, MMP 还提供了可以提取多个元素的函数指令, 包括提取行、列向量以及子矩阵的函数. 下面还是以矩阵 A 为例, 先来看提取行、列的函数:

```
>row(A, 2);
```

$[4,5,6]$

```
>row(A, 1, 2);
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

```
>col(A, 3);
```

$$\begin{bmatrix} 3 \\ 6 \\ 9 \end{bmatrix}$$

```
>col(A, 2, 3);
```

$$\begin{bmatrix} 2 & 3 \\ 5 & 6 \\ 8 & 9 \end{bmatrix}$$

函数 row 和 col 可以用于提取矩阵的单行和单列, 或者多行和多列. 当提取多行和多列时, 函数的后两个整数参数分别代表行的范围和列的范围, 这意味着被提取的行或列必须是连续排列的. 另外需要注意的是, 这两个函数都把提取出的行或列构造成一个新的矩阵返回.

现在再来看提取子矩阵的函数 submatrix. 为了使用起来更加方便灵活, MMP 为该函数提供多种调用形式, 用户可以根据需要进行选择.

```
>submatrix(A, 1, 2, 2, 3);
```

$$\begin{bmatrix} 2 & 3 \\ 5 & 6 \end{bmatrix}$$

```
>submatrix(A, [3, 1, 2], [1, 3]);
```

$$\begin{bmatrix} 7 & 9 \\ 1 & 3 \\ 4 & 6 \end{bmatrix}$$

```
>submatrix(A, [3, 1, 2], 1, 3);
```

$$\begin{bmatrix} 7 & 8 & 9 \\ 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

```
>submatrix(A, 2, 3, [3, 1]);
```

$$\begin{bmatrix} 6 & 4 \\ 9 & 7 \end{bmatrix}$$

上面给出了 submatrix 的四种形式提取 A 的子矩阵. 第一个参数表示被操作的矩阵, 其余参数都是整数或整数的链表, 它们分别表示行的范围或行的列表, 以及列的范围或列的列表, 读者可以清楚地分辨每种形式的用法的意义. 以第四条指令为例, 它表示提取矩阵的第 2 行到第 3 行的元素, 再从这些行里提取第 3 列和第 1 列的元素, 做成一个矩阵. 可以看到, 在用 submatrix 提取子矩阵时, 并不要求提取出的行或列必须相邻, 而且行或列的位置关系也可以发生变化.

在上面使用函数 submatrix 的第二条指令中, 要求提取的行和列都以列表的形式给出, 事实上, 这种调用形式具有最普遍的意义, submatrix 的其他调用形式, 以及前面所介绍的函数 row 和 col, 都可以看作是这种调用形式的特例.

前面讲过修改矩阵中的单个元素的方法, MMP 还提供了另外一些修改矩阵的函数, 例如删除矩阵的行或列等. 这些函数将使矩阵的形状和维数发生变化.

```
>delrow(A, 1);
```

$$\begin{bmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```
>delcol(A, 1);
```

$$\begin{bmatrix} 2 & 3 \\ 5 & 6 \\ 8 & 9 \end{bmatrix}$$

```
>A;
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

函数 delrow 和 delcol 分别删除矩阵的某一行和某一列. 请注意, 这两个函数将返回删除矩阵的某行或某列之后构造的新矩阵, 原矩阵仍保持不变.

3. 基本矩阵运算. 矩阵的基本运算包括矩阵的加 (减) 法、乘法、数乘和幂等等. 下面将以本节最开始定义的两个矩阵 A 和 B 之间的运算为例, 展示 MMP 中矩阵的这些基本运算的使用方法.

```
>A+B;
```

$$\begin{bmatrix} 1+a & 2+b & 3+c \\ 4+d & 5+e & 6+f \\ 7+g & 8+h & 9+i \end{bmatrix}$$

```
>A*B;
```

$$\begin{bmatrix} a+2d+3g & b+2e+3h & c+2f+3i \\ 4a+5d+6g & 4b+5e+6h & 4c+5f+6i \\ 7a+8d+9g & 7b+8e+9h & 7c+8f+9i \end{bmatrix}$$

```
>x*A+x^2*B;
```

$$\begin{bmatrix} x+x^2a & 2x+x^2b & 3x+x^2c \\ 4x+x^2d & 5x+x^2e & 6x+x^2f \\ 7x+x^2g & 8x+x^2h & 9x+x^2i \end{bmatrix}$$

对矩阵做算术运算, 最简单而且直观的方法就是直接使用运算符. 从以上示例中可以看到, 矩阵的算术运算符和单个数据元素的完全相同. 在前面已经提到过, MMP 对矩阵类型的对象采取 “自动求值” 的机制. 该机制在应用于矩阵的运算时, 表现为系统可以直接计算出矩阵的算术表达式的结果, 而不像 Maple 系统那样需要强制地调用矩阵的求值函数. 另外需要说明的是, 在 MMP 中, 运算符 “*” 既可以用作矩阵的乘法, 也可以用作矩阵的数乘. 可以看到, 当运算符 “*” 被用作简单数据类

型的单个数据元素之间的乘法运算符时, 相应的运算满足交换律, 而如果用它进行矩阵的乘法运算, 这个运算并不满足交换律. 也就是说, 在 MMP 中, 运算符 “*” 的性质是由运算数的数据类型决定的. 当运算数的类型不同时, 相同的运算符可能具有不同的含义.

MMP 也可以对矩阵做乘幂运算, 其乘幂运算符 “∧” 也和单个数据元素的相同. 当然, 我们只能对方阵进行乘幂运算. 例如：

```
>A^3;
```

$$\begin{bmatrix} 468 & 576 & 684 \\ 1062 & 1305 & 1548 \\ 1656 & 2034 & 2412 \end{bmatrix}$$

```
>B^2;
```

$$\begin{bmatrix} a^2+bd+cg & ab+be+ch & ac+bf+ci \\ da+ed+fg & bd+e^2+fh & dc+ef+fi \\ ga+hd+ig & gb+he+ih & cg+fh+i^2 \end{bmatrix}$$

对于矩阵的基本运算, 除了上面展示的直接使用运算符的实现方式外, MMP 也提供了一套相应的函数来完成这些基本运算, 以方便用户在不同场合下选择合适的使用方式. 相关的函数有：“matadd: 矩阵相加”, “matminus: 矩阵相减”, “multiply: 矩阵相乘”, 以及 “matexp: 矩阵乘幂”. 可以通过这些函数同样完成前面例子中的基本运算：

```
>matadd(x*A, x^2*B);
```

$$\begin{bmatrix} x+x^2a & 2x+x^2b & 3x+x^2c \\ 4x+x^2d & 5x+x^2e & 6x+x^2f \\ 7x+x^2g & 8x+x^2h & 9x+x^2i \end{bmatrix}$$

```
>multiply(A, B);
```

$$\begin{bmatrix} a+2d+3g & b+2e+3h & c+2f+3i \\ 4a+5d+6g & 4b+5e+6h & 4c+5f+6i \\ 7a+8d+9g & 7b+8e+9h & 7c+8f+9i \end{bmatrix}$$

```
>matexp(A, 3);
```

$$\begin{bmatrix} 468 & 576 & 684 \\ 1062 & 1305 & 1548 \\ 1656 & 2034 & 2412 \end{bmatrix}$$

以上函数的运算结果与前面例子中直接使用运算符时是完全相同的, 调用方法也十分简单. 只是在使用函数 multiply 时需要注意两个矩阵的顺序.

4. 其他矩阵运算. 前面介绍了矩阵的基本运算. 下面将进一步介绍矩阵的其他相关运算, 这些运算为用户提供了更加全面的线性代数计算工具.

行列式在许多线性代数问题中都有着广泛的应用, 它是计算矩阵的特征值与特征向量的基础. MMP 的函数 det 计算矩阵的行列式. 注意, 行列式计算只能对方阵进行. 例如：

```
>det(A);
0
>det(B);
a e i - a f h - d b i + d c h + g b f - g c e
>C:=matrix(3, 3, [x, y, z, 2*x, y, 0, 0, 2*x, y]):
>det(C);
```

$4zx^2 - y^2x$

假设矩阵 A 为 $m \times n$ 阶矩阵, 将 A 的行、列互换而得到的 $n \times m$ 阶矩阵 B 称为 A 的 *转置矩阵*, 记为 $B = A^T$. 函数 tran 实现对矩阵的转置, 例如：

```
>M:=matrix(3, 2, [1, 0, 4, 1, 2, 3]):
>N:=matrix(2, 3, [1, 2, 3, 4, 5, 6]):
>multiply(tran(N), tran(M));
```

$$\begin{bmatrix} 1 & 8 & 14 \\ 2 & 13 & 19 \\ 3 & 18 & 24 \end{bmatrix}$$

```
>tran(multiply(M, N));
```

$$\begin{bmatrix} 1 & 8 & 14 \\ 2 & 13 & 19 \\ 3 & 18 & 24 \end{bmatrix}$$

以上例子中利用转置函数 tran 和乘积函数 multiply 验证了转置的一个性质：$(\mathrm{M} * \mathrm{N})^T = \mathrm{N}^T * \mathrm{M}^T$.

假设 A 和 B 为两个方阵, 若它们满足 $A * B = I$(这里的 I 为单位矩阵), 则矩阵 B 被称为矩阵 A 的 *逆矩阵*, 记为 $B = A^{-1}$. 计算逆矩阵是线性代数中的基本问题之一. MMP 提供了函数 inverse 来进行逆矩阵的计算. 看下面例子：

```
>F:=matrix(3, 3, [1, 0, 2, 0, 4, 1, 2, 4, 2]):
>FI:=inverse(F);
```

$$\begin{bmatrix} -1/3 & -2/3 & 2/3 \\ -1/6 & 1/6 & 1/12 \\ 2/3 & 1/3 & -1/3 \end{bmatrix}$$

```
>multiply(F, FI);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

MMP 除了能对数值矩阵求逆矩阵外, 还能计算符号矩阵的逆矩阵, 例如：

```
>inverse(matrix (2, 2, [a, b, c, d]));
```

$$\begin{bmatrix} -\frac{d}{-da+bc} & \frac{b}{-da+bc} \\ \frac{c}{-da+bc} & -\frac{a}{-da+bc} \end{bmatrix}$$

从线性代数理论可知, 一个矩阵可逆的充分必要条件是该矩阵的对应行列式的值不为 0. 在 MMP 中, 如果对不可逆矩阵调用函数 inverse, 系统将返回错误信息. 例如, 前面定义的矩阵 A 的行列式的值恰好为 0, 因此它是一个不可逆矩阵.

```
>det(A);
0
>inverse(A);
Error! This is a singular matrix!
```

5. 线性方程组求解. 线性方程组求解问题是线性代数中基本而重要的问题之一. 众所周知, 求解线性方程组的最简单而且最基本的方法是 Gauss 消元法及其变形. Gauss 消元法的基本思想是利用矩阵的初等变换, 将方程组的增广矩阵转化为上三角矩阵, 然后通过回代获得方程组的解. 矩阵的初等变换则是 Gauss 消元法的基础, 当然也是求解线性方程组的基础. 矩阵的初等变换包括初等行变换和初等列变换两种. 以初等行变换为例, 它又包括以下三种变换形式：

1. 用一个非零的常数乘以矩阵的某一行.
2. 交换矩阵中的任意两行.

3. 将矩阵中某一行的倍数加到另一行.

以上是三种针对矩阵行向量进行操作的初等行变换, 与此相对应的, 还有三种针对列向量进行操作的初等列变换, 这里就不再赘述. 作为实现 Gauss 消元法的辅助函数, 也为了便于用户一步步地推导线性方程组的求解过程, MMP 提供了对应于每一种初等变换的函数指令, 这些函数可以帮助用户更加深入、直观地理解各种消元法的实质. 作为例子, 我们来看第二种初等变换对应的函数指令. 为了实现矩阵中任意两行或两列的互换, MMP 提供了两个相应的函数 swaprow 和 swapcol. 在以下例子中, 矩阵 A 仍旧沿用在本节最开始时的定义：

```
>swaprow(A, 1, 2);
```

$$\begin{bmatrix} 4 & 5 & 6 \\ 1 & 2 & 3 \\ 7 & 8 & 9 \end{bmatrix}$$

```
>swapcol(A, 1, 2);
```

$$\begin{bmatrix} 2 & 1 & 3 \\ 5 & 4 & 6 \\ 8 & 7 & 9 \end{bmatrix}$$

MMP 提供了两个线性方程组求解函数 lsolve 和 linsolve, 用户可以直接利用这两个函数方便地获得线性方程组的解. 需要注意的是, 函数 lsolve 和 linsolve 的调用方式有所不同. 下面通过一个简单例子说明这两个函数的不同用法.

例 2.7.1 *解方程组*

$$\begin{cases} x_1 + 2x_2 + 3x_3 - 14 = 0 \\ x_1 + 3x_2 + 4x_3 - 19 = 0 \\ x_1 + 4x_2 + 2x_3 - 15 = 0 \end{cases}$$

我们可以按照下面的方法求解：

```
>lsolve([x1+2x2+3x3-14, x1+3x2+4x3-19, x1+4x2+2x3-15],
        [x1, x2, x3]);
[1, 2, 3]
```

函数 lsolve 的第一个参数是方程组等号左边的多项式所组成的链表, 第二个参数是由待求解的未知数所组成的链表. lsolve 返回的结果是由求得的未知数的解所组成的链表. 注意, 解的顺序与第二个参数中给定的未知数的顺序一致.

在线性代数中, 人们通常把线性方程组转化为与其等价的矩阵问题去求解. MMP 也提供了以矩阵形式输入的线性方程组求解函数, 这就是函数 linsolve. 同

样求解例 2.7.1 中的线性方程组, 我们首先把它表示成矩阵的形式：

$$\begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 4 \\ 1 & 4 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 14 \\ 19 \\ 15 \end{pmatrix}$$

其增广矩阵为

$$\begin{pmatrix} 1 & 2 & 3 & 14 \\ 1 & 3 & 4 & 19 \\ 1 & 4 & 2 & 15 \end{pmatrix}$$

函数 linsolve 只有一个参数, 就是所求解的线性方程组的增广矩阵. 现在已经可以按照如下方法完成对方程组的求解了：

```
>m:=matrix(3, 4, [1, 2, 3, 14, 1, 3, 4, 19, 1, 4, 2, 15]):
>linsolve(m);
[1, 2, 3]
```

不难看出, 函数 lsolve 与 linsolve 针对同一个线性方程组求解的输出形式是相同的, 只是输入的参数形式有所不同, 用户可以根据不同场合的需要任意选择其中之一进行求解.

下面看一些不同情形下的线性方程组求解的例子.

例 2.7.2　*解方程组*

$$\begin{cases} 3x_1 - 5x_2 + 7x_3 + x_4 = 1 \\ 2x_1 - x_2 + 4x_3 + 3x_4 = 1 \\ x_1 + 5x_2 + 5x_3 + 6x_4 = 1 \\ -2x_1 + 3x_3 + 7x_4 = 1 \end{cases}$$

```
>m:=matrix(4,5,[3,-5,7,1,1,2,-1,4,3,1,1,5,5,6,1,-2,0,3,7,1]):
>linsolve(m);
```

$$\left[\frac{11}{57}, \frac{-9}{133}, \frac{-1}{57}, \frac{82}{399}\right]$$

不难发现例 2.7.1 和例 2.7.2 中线性方程组的共同点, 即它们的系数矩阵都是方阵且行列式不为零. 这样的线性方程组一定有唯一解, 这就意味着在保持系数矩阵不变的前提下, 等号右边的系数取任意值时, 该线性方程组都应该保证有唯一解. 在这种情况下, 如果把本例等号右边的系数改为符号, linsolve 仍然能够对它进行求解.

例 2.7.3 解方程组

$$\begin{cases} 3x_1-5x_2+7x_3+x_4=a \\ 2x_1-x_2+4x_3+3x_4=b \\ x_1+5x_2+5x_3+6x_4=c \\ -2x_1+3x_3+7x_4=d \end{cases}$$

```
>m:=matrix(4,5,[3,-5,7,1,a,2,-1,4,3,b,1,5,5,6,c,-2,0,3,7,d]):
>linsolve(m);
```

$$\left[\frac{-16a+45b-7c-11d}{57},\frac{a-23b+23c-10d}{133},\frac{17a-30b+11c+d}{57},\frac{-83a+180b-47c+32d}{399}\right]$$

下面我们看一个系数矩阵是方阵且行列式为零（即系数矩阵不满秩）的例子. 我们知道, 这样的线性方程组可以有无穷多个解, 通过函数 linsolve 可以求得其通解.

例 2.7.4 解方程组

$$\begin{cases} x_1+10x_2+15x_3=18 \\ 4x_1+25x_2+42x_3=51 \\ 7x_1+40x_2+69x_3=84 \end{cases}$$

```
>m:=matrix(3,4,[1,10,15,18,4,25,42,51,7,40,69,84]):
>linsolve(m);
```

$[-3x_3+4,-6/5x_3+7/5,x_3]$

以上求解结果表明, x_3 是自由变量, x_1 和 x_2 被分别表示成关于这个自由变量的表达式. linsolve 给出了该方程组的通解.

函数 linsolve 不仅可以用来求解上述系数矩阵为方阵的线性方程组, 而且可以对系数矩阵不是方阵的线性方程组进行求解. 这样的方程组也可以有无穷多个解, 如下例所示.

例 2.7.5 解方程组

$$\begin{cases} x_1+3x_2+3x_3+2x_4=-1 \\ 2x_1+6x_2+9x_3+5x_4=4 \\ -x_1-3x_2+3x_3=13 \end{cases}$$

```
>m:=matrix(3,5,[1,3,3,2,-1,2,6,9,5,4,-1,-3,3,0,13]):
>linsolve(m);
```

$$\left[-3x_2-x_4-7,x_2,\frac{-x_4+6}{3},x_4\right]$$

这个方程组有 3 个方程, 4 个未知数, 即系数矩阵不是方阵. linsolve 的求解结果表明, x_2 和 x_4 是自由变量, x_1 和 x_3 被分别表示成关于这两个自由变量的表达式. 也就是说, 在这种情况下, linsolve 也给出了方程组的通解.

到目前为止, 看到的例子都是线性方程组有解的情形, 下面看一个无解的例子.

例 2.7.6 *解方程组*

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 1 \\ 4x_1 + 5x_2 + 6x_3 = 2 \\ 7x_1 + 8x_2 + 9x_3 = 4 \end{cases}$$

```
>m:=matrix(3,4,[1,2,3,1,4,5,6,2,7,8,9,4]):
>linsolve(m);
The equation has no solution!
```

这是一个系数矩阵的行列式为零 (读者可自行验证), 线性方程组无解的例子. 我们看到, 当方程组无解时, 系统会给出提示信息.

除了运用传统的 Gauss 消元法求解线性方程组以外, MMP 还引进了一个关于 Gauss 消元法的 Bareiss 改进方法, 即所谓 Bareiss 消元法. 理论研究表明, 采用 Bareiss 消元法, 可以有效地解决 Gauss 消元法在求解过程中经常带来的中间系数膨胀的问题, 从而降低了计算的复杂度, 提高了运算效率. 实验表明 Bareiss 消元法尤其是对系数矩阵的阶数比较大的线性方程组, 以及具有符号系数的线性方程组具有明显的效率方面的优势. 由于篇幅所限, 这里不准备进一步讨论采用 Bareiss 消元法的具体求解过程. 下面再看两个符号线性方程组求解的例子.

例 2.7.7 *解方程组*

$$\begin{cases} ax + 2y - 7z = 5 \\ 4x + by - 18z = -33 \end{cases}$$

```
>m:=matrix(2,4,[a,2,-7,5,4,b,-18,-33]):
>linsolve(m);
```

$$\left[\frac{(7x_3b + 5b - 36x_3 + 66)}{(ba - 8)}, \frac{(18x_3a - 33a - 28x_3 - 20)}{(ba - 8)}, x_3\right]$$

例 2.7.8 *解方程组*

$$\begin{cases} ax_1 + (b-1)x_2 + (a-1)x_3 = -a \\ 2ax_1 + (2b-2)x_2 + (2a+b)x_3 = ab \end{cases}$$

```
>m:=matrix(2,4,[a,b-1,a-1,-a,2*a,2*b-2,2*a+b,a*b]):
>linsolve(m);
```

$$\left[\frac{(-bx_2 + x_2 - a^2)}{(a)}, x_2, a\right]$$

§2.8 op 与 subs 函数

本节介绍两个 MMP 的基本函数：op 与 subs. 它们的共同特点是, 可以作用于 MMP 的多个数据类型. 由于这两个函数在 MMP 编程中有着非常广泛的应用, 所以我们单独放在一节中介绍.

1. op 函数. op 函数将一个复杂的数据类型分解, 给出其组成部分. 这一函数在 MMP 编程中起着重要作用.

限于篇幅, 下面我们只介绍 op 作用于多项式的情况. op 作用于其他数据类型的情况读者可参阅《MMP 用户手册》. 设 P 是一个多项式, 将其写成单项式之和的形式：

$$P = c_0M_0 + c_1M_1 + \cdots + c_sM_s \tag{2.2}$$

如果 $s \geqslant 1$, 则 op(P) 返回如下列表：

$$[c_0M_0, c_1M_1, \cdots, c_sM_s]$$

如果 $P = c\prod_{i=1}^{n} x_i^{d_i}$ 是一个单项式, 并且 $c \neq 1$ 或者 $c = 1$ 且至少有两个 $d_i > 0$, 则 op(P) 返回如下列表：

$$[c, x_{k_1}^{d_{k_1}}, \cdots, x_{k_s}^{d_{k_s}}], \text{ 其中 } d_{k_i} > 0, i = 1, \cdots, s.$$

如果 $P = x_i^{d_i}$, 则 op(P) 返回列表 $[x_i, d_i]$.

下面是一些例子.

```
>op(2*x1^3+x1*x2^2+x2^2);
```
$[2x_1^3, x_1x_2^2, x_2^2]$

```
>op(2*x1^3);
```
$[2, x_1^3]$

```
>op(x1*x2^2);
```
$[x_1, x_2^2]$

```
>op(x2^2);
```
$[x_2, 2]$

2. subs 函数. subs 可以对表达式, 多项式, 有理多项式中的变元进行替换. 它有如下几种用法：

```
subs(f,v,e); subs(fs,v,e); subs(f,vs,es); subs(fs,vs,es).
```

首先介绍第一种形式 subs(f,v,e). 其中 f 是待替换的公式, v 是将要替换的变元, e 是该变元要替换的公式. subs 将公式 f 中的变元 v 替换为公式 e. 这里 f 和 e 可

以是任意数据类型. 替换的结果的数据类型将根据计算结果而定. subs 的运算步骤如下:

- 如果 f 是列表或矩阵, 则在 f 的元素中递归寻找含有变量 v 的表达式、有理式、多项式或变量 v 本身.
- 根据下面数据类型的顺序, 将 f 或 e 变为顺序高的一种.

$$\text{表达式} > \text{有理式} > \text{多项式} > \text{变量}$$

- 将 f 中的 v 替换为 e. 注意, 我们只做一遍替换. 否则, 如果 e 中含有 v, 则会产生死循环.

subs 其他形式与第一种形式类似. 其中 fs 是待替换公式的列表, vs 是将要替换的变元的列表, es 则是 vs 中相应变元要替换的公式的列表. 下面是一些例子.

```
>subs(x^2+y, x, t);
```

$y + t^2$

```
>subs(x^2+y, [x,y], [2,m]);
```

$m + 4$

```
>subs([x^2+y, p/x], x, 2);
```

$\left[y + 4, \dfrac{p}{2}\right]$

```
>subs([x^2+y, p/x], [x, p], [2, t]);
```

$\left[y + 4, \dfrac{t}{2}\right]$

第三章　MMP 的编程环境

在前面两章里, 我们一直把 MMP 看作是许多 “黑箱命令” 的汇集, 使用它内部定义的符号计算功能. 实际上 MMP 还是一种高级程序设计语言. 由于 MMP 承担的大多数任务是 “一次性” 的计算, 所以我们将 MMP 语言设计为解释性语言而非编译语言. MMP 语言可分成两部分：一般语句和自定义函数. 一般语句在 MMP 的用户界面上直接解释执行; 自定义函数使得用户可以先定义好自己所需的函数, 待以后实际需要做相应的计算时再调用它们. 本章介绍 MMP 编程语言与用户自定义函数.

§3.1　介　　绍

1. 基本数据类型.　MMP 提供了多种基本的数据类型, 有关情况已经在第二章第一节详细介绍, 这里不再赘述. 可用 MMP 的 type 命令询问任何表达式的类型. 这一功能对于识别复杂的类型结构特别有用. type 命令实际已在第二章介绍过, 这里不妨再看几个例子.

```
>x := 1:
>type(x);
Number(BigNumber)
>x := (y+1)^2 + 1:
>type(x);
Expression
>type(expand(x));
Polynomial
```

2. 保留字.　同其他所有语言一样, MMP 也有自己的保留字. 这些保留字是系统保留的, 用户的变量命名、函数命名均不能与这些保留字冲突. MMP 的保留字见表3.1.

值得一提的是 restart 和 quit 命令.　restart 命令清除本次执行期中已经定义的所有变量及函数, 使系统恢复到初始状态; 而 quit 命令使 MMP 退出执行.

```
>x:=1:
>x;
```

表 3.1　MMP 的保留字

and	or	not	if	then	elif
else	fi	for	from	by	to
while	do	od	break	continue	proc
local	global	return	end	restart	quit

```
1
>restart;
>x;
x
>quit;
```

在下面的介绍中严格地给出各种结构的语法形式. 其中

- 顺序出现的一系列元素表示程序里相应的部分应该顺序出现;
- 用一对尖括号括起的 $\langle\cdot\rangle$ 代表相应的结构, 例如 $\langle simple\ expr\rangle$;
- 打字机体的文字是 MMP 语言的关键字;
- 符号 | 表示 “或”, 说明有关结构 (或结构中的部分) 可以具有几种不同的形式. 在语法描述中, | 具有最低的优先级. 例如对于 $a\ b \mid c$ 表示该结构的形式可以是 $a\ b$ 或者 c;
- 圆括号用于表示分组, 例如 $a\ (b \mid c)$ 表示该结构可以是 $a\ b$ 或者 $a\ c$;
- 方括号表示其中内容为可选, 例如 $a\ [b]$ 表示该结构可为 a 或者 $a\ b$;
- 方括号后跟 + 表示其中内容的一次或者多次重复出现, 例如 $a\ [b]+$ 表示该结构可以是 $a\ b$、$a\ b\ b$、$\cdots$;
- 方括号后跟 $*$ 表示其中内容可以没有, 也可以一次或多次重复出现, 例如 $a\ [b]*$ 表示该结构可以是 a、$a\ b$、$a\ b\ b$、$\cdots$;
- 其他字符表示在 MMP 程序里应该出现的字符.

§3.2　基 本 语 句

对用户而言, MMP 就像是一个交互式计算器. 用户通过界面输入所需的命令, MMP 分析所得到的输入, 如果没有语法错误则执行遇到的命令并返回计算的结果. 如果输入中有语法错误, MMP 就会拒绝这个命令, 并输出相应的错误信息. 这一过程不断地重复进行, 直至用户输入一个 quit 命令.

1. 关系表达式.

语法：

$\langle simple\ expr\rangle\ \langle relation\ op\rangle\ \langle simple\ expr\rangle$

说明：

两个表达式之间的大小关系可以用关系运算符确定, 大于号 $>$, 小于号 $<$, 等于号 $=$, 大于等于号 $>=$, 小于等于号 $<=$, 不等于号 $<>$. 关系运算符具有相同的优先级, 它们的优先级低于所有的算术运算符. 关系表达式的值是 true 或 false.

例子：

```
>1 < 2;
true
```

2. 逻辑表达式.

语法：

$\langle relation\ expr\rangle$ [(`and` | `or`)$\langle relation\ expr\rangle$] +
　　| `not` $\langle relation\ expr\rangle$

说明：

逻辑运算符用于描述复杂的逻辑关系, 可以用于组合两个或者多个关系表达式表示的逻辑关系, 也可以用于组合逻辑表达式本身. 如 $1 < 2$ *and* $2 < 3$, $1 < 2$ *and* $2 < 3$, *not* $1 < 2$. 在使用这些运算符时, 必须特别关注运算符之间的优先级关系, 必要时自己加括号控制计算的进程. 逻辑运算符的优先级从高到低顺序为 *not* $>$ *and* $>$ *or*, 逻辑运算符的优先级低于所有关系运算符. 逻辑表达式的值是 true 或 false.

例子：

```
>1>2 and 1<2 or 1<2;
true
>not 1>2 or 1<2;
true
>i:=1: j:=2:
>if (not i>2 and j<=2) then s:=x^2+1 end if;
s:=x^2+1
```

3. 赋值语句.

语法：

$\langle variable\rangle\ :=\ \langle simple\ expr\rangle$

说明：

赋值语句可能是 MMP 中使用最多的语句. MMP 的编程语言是一种弱类型的语言, 变量使用前无需事先定义. 也就是说, $\langle variable\rangle$ 是没有类型的, 变量的值具有特定

的数据类型. 进一步说, 即使一个变量当时的值是某个类型, 也完全可以用另一类型的数据给它赋值.

例子：

```
>f:=321321:
>type(f);
Number(BigNumber)
>f:=x^2+3:
>type(f);
Expression
```

另外, 赋值语句所隐含的类型变化特别值得我们注意, 例如：

```
>y:=1:
>y[0]:=1: y[x]:=2:
```

第一个语句把一个整数类型的值赋给 y, 但是当 MMP 执行 y[0] := 1 时, y 的值被隐含地改变为 table 类型. table 类型的数据可以看成一种两列的表 (表格), 其中的第一列是其索引, 第二列是与索引对应的值. 对于表而言, 合法的操作只有索引操作, 直接引用 y 是类型错误, 例如：

```
>t:=y^2+1;
Exp Error: Wrong Type!!!!
Error: run time error, may partly execute
>t:=y:
>t[0];
1
```

当赋值语句以冒号结束时, 不显示任何结果；当赋值语句以分号结束时, MMP 计算出赋值运算符右边的表达式的值并将该赋值语句显示出来.

```
>a:=2: b:=3:
>c:=a^b;
c:=8
```

4. 选择语句.

语法：

if ⟨*conditional expr*⟩ then ⟨*statement list*⟩

[elif ⟨*conditional expr*⟩ then ⟨*statement list*⟩]∗

[else ⟨*statement list*⟩]

(end if | fi)

说明:

选择语句提供了根据条件 (⟨*conditional expr*⟩) 选择执行语句的能力. 结构中的 `elif` ⟨*conditional expr*⟩ `then` ⟨*statement list*⟩ 部分可以不出现, 也可以重复多次. 关键字 `elif` 代表 `else if`. 在执行时, MMP 依次检测 `if`、`elif` 之后各个条件的真假, 在第一次遇到条件为真时, 就执行相应的语句, 并跳出这一选择语句; 如果所有条件都不为真, 那么执行 `else` 对应的语句. 在实际应用中, 可以利用 `and`、`or` 等逻辑运算符描述复杂的判断条件. 当 if 语句以冒号结束时, 不显示任何结果; 当 if 语句以分号结束时, 显示所执行的分支中的每个语句的运行结果. end if, fi, else, elif 等关键字之前的最后一个语句可以没有结束符号.

例子:

```
>f := 1:
>if f<2 then g:=f+1 else g:=f+2 end if;
g:=2
>if (g<f) then g:=g+1 elif (g=f+1) then g:=g+2 else g:=g+3 fi;
g:=4
>if igcd(4, 7)=1 and igcd(4, 8)<>1 then result:=1 fi;
result:=1
```

5. for 循环语句.

语法:

`for` ⟨*variable*⟩ `from` ⟨*simple expr*⟩ [`by` ⟨*simple expr*⟩] `to` ⟨*simple expr*⟩
  `do` ⟨*statement list*⟩
(`end do` | `od`)

说明:

循环语句提供重复执行循环体内语句序列 ⟨*statement list*⟩ 的能力. MMP 首先检测 ⟨*variable*⟩ 的值是否大于 `to` 之后的表达式的值, 如果为否, 则执行作为循环体的 ⟨*statement list*⟩. 执行完循环体之后, 将 ⟨*variable*⟩ 的值增加 `by` 之后的表达式的值. 当 ⟨*variable*⟩ 的值大于 `to` 之后的表达式的值时, 这个 for 语句停止执行. 当 `by` ⟨*simple expr*⟩ 缺省时, 默认的增量为 1. 当 for 语句以冒号结束时, 不显示任何结果; 当 for 语句以分号结束时, 显示循环体内每个语句在每次执行时的运行结果. end do 或 od 之前的最后一个语句可以没有结束符号.

例子:

```
>a:=0:
>for i from 1 by 2 to 5 do a:=a+1 end do;
a:=1
```

```
a:=2
a:=3
>i;
7
>a;
3
>for i from 1 to 2 do 121 od;
121
121
>for i from 1 by -1 to -4 do 1 end do:
>i;
-5
```

6. while 循环语句.

语法:

while ⟨*conditional expr*⟩
    do ⟨*statement list*⟩
(end do | od)

说明:

while 语句的功能与 for 语句类似. 它也提供重复执行的能力, 但是根据一个逻辑条件确定是否继续. 在执行 while 语句时, MMP 首先检测条件 ⟨*conditional expr*⟩ 的真假, 若条件为真, 则执行作为循环体的 ⟨*statement list*⟩ 一次. 执行完后重复上述检测过程. 当循环条件 ⟨*conditional expr*⟩ 为假时停止. 当 while 语句以冒号结束时, 不显示任何结果; 当 while 语句以分号结束时, 显示循环体内每个语句在每次执行时的运行结果. end do 或 od 之前的最后一个语句可以没有结束符号.

例子:

```
>a := 5:
>while (a>=1) do a:=a-1 end do:
>a;
0
```

7. break, continue 语句.

语法:

```
break
continue
```

说明:

break 语句用于使执行进程跳出最内层的循环语句, 转去执行这个循环之后的下一语句; continue 语句的作用是中止此次循环体的执行, 继续这一循环的下次执行. break 和 continue 语句均只能用于循环语句中. 如果在非循环语句中使用了这两个语句, MMP 系统认为这是一个语义错误, 而非语法错误. MMP 的语法检查并不检查这种错误, 因此这样的语句可能部分执行, 直到 MMP 发现这一语义错误时中止.

例子:

```
>a:=1: p:=0:
>while (a<5) do
    a:=a+1;
    if (a=2) then igcd(6, 9)
    elif (a=3) then continue
    else break
    end if;
    p:=p+1
 end do:
>p;
1
>if 1<2 then a; break; p end if;
4
Error:run time error, may partly execute
```

8. return 语句.

语法:

`return` [⟨*simple expr*⟩]

说明:

return 语句用于从函数中返回. 如果写不带值部分的 return 语句, 就表明有关函数并不返回值, 这种函数的调用不能作为 r-value; 需要返回值的函数里必须使用带表达式的 return 语句. 在执行这种语句时, 表达式 ⟨*simple expr*⟩ 的值就成为函数调用的值. return 语句的例子在下一节里讨论自定义函数时一起介绍.

9. 注释.

语法:

\# ⟨*comments*⟩

说明:

实践说明, 在编写程序时, 一个很好的习惯就是给程序加适当的注释. 注释的出现并不改变程序的意义, 它只是提供一些供人阅读的信息, 以便帮助人更好的理解这一程序的意义. 注释以符号 `#` 开始, 直到这一行的结束.

例子:

```
>#this line is an example of comments in program
 f:=0;
 f:=0
```

§3.3　表

在本书里, 我们把表 (table) 类型作为单独的一节讨论, 主要是因为在 MMP 编程中, 表类型的使用非常频繁. 下面我们直接用英文 table 代替表. 一个 table 类型的数据对象, 也就是一个映射 $y = f(x)$, 其中的 x 是这个 table 的索引, $f(x)$ 就是该索引所对应的值. 一个索引有一个对应的值, 也就是说, 该映射是一一映射.

1. 表的定义. MMP 并没有为 table 类型单独提供专门的关键字或函数, 一旦程序里对某个变量的某个索引进行赋值, MMP 就会自动将该变量看作具有一个 table 类型的值, 并为给定的索引关联相应的值. 例如:

```
>#y isn't defined as table type before
 y[0]:=x^2+x+3:
```

在这一命令之前, 并不需要任何有关 y 的值属于 table 类型的说明, 而当我们输入 y[0]:=$\mathrm{x}^2+\mathrm{x}+3$ 时, MMP 就为 y 建立了一个 table 类型的数据对象, 并将 y 的索引 0 映射到 $\mathrm{x}^2+\mathrm{x}+3$. 如果在随后输入:

```
>#y has been defined as table type before
 y[1.2]:=3:
```

此时 MMP 做的就是为 y 添加另外一条映射, 把索引 1.2 映射到整数 3.

2. 表的索引. 原则上说, 索引可以是任何的值. 实际中使用最多的是类似数组的整数下标. 因此可以用 table 类型的数据对象代替数组, 当然 table 类型不会像数组类型那样对引用下标进行越界检查.

```
>#x undefined before
 y[x]:=2:
>y[x];
2
>for i from 1 to 10 do x[i]:=i^2 end do:
```

在第一个赋值语句中, 变量 y 的索引为变量 x, 相关联的值是整数 2. 在 for 循环语句中, 变量 x 的索引则是整数 1 到 10, 此时 x 相当于是一个数组.

3. 表的使用.　表的使用非常简单. 如果被引用的索引在此之前赋过值时, 索引表达式就返回该索引所对应的值; 如果该索引以前没有赋过值, MMP 就把它看作一个未定义的变量.

```
>y[0]:=x^3+2*x+1: y[1]:=diff(y[0], x):
>p := y[0]+y[1]+y[2]^2;
p:=3*x^2+2+y[2]^2+x^3+2*x+1
```

在 § 3.5 中, 我们将用一个比较大的程序来说明 MMP 编程. 那里的很多函数例都用到了 table 类型, 通过那些例子, 读者可以掌握 table 的使用方式.

§3.4　自定义函数

如果在使用 MMP 时, 需要实现一个相对独立的功能, 或者在工作中经常需要运行一些相同的命令序列, 我们就可以考虑把它们定义为函数. 一旦定义好这种函数, 在需要时, 就可以采用简单的语法形式直接调用它们. 可以考虑定义为函数的例子如排序, 求特征列等. 这一节将详细介绍在 MMP 中自定义函数的编写及调用.

1. 语法规则.

$\langle function\ name\rangle$:= `proc` ([$\langle params\ list\rangle$])
[`local` $\langle variables\ declaration\rangle$]
[`global` $\langle variables\ declaration\rangle$]
　　[$\langle statement\ list\rangle$]
`end`[`proc`]

`proc` 关键词表明这里写的是一个函数定义. $\langle function\ name\rangle$ 是被定义函数的名字, $\langle params\ list\rangle$ 是函数的形式参数列表, 可以为空. MMP 统一采用值传递的方式, 因此函数内部对参数的改变不影响传入的变量的值. `global` 和 `local` 是域声明符. 这里要求具有相同域特征的变量必须在一个声明里描述. 也就是说, 不能出现诸如 “global x, y; global z;” 的情况, 这些变量的声明必须放在一起写为 “global x, y, z;”. 在一个函数里, 允许把一部分变量定义为 `global`, 另一部分变量定义为 `local`. $\langle statement\ list\rangle$ 是函数体, 它也可以为空. 最后的 `end` 或 `end proc` 表明函数定义结束.

从某种角度看, 函数定义也可以看成是一种赋值语句, 可以认为其定义形式 f := `proc`([$\langle parmas\ list\rangle$]) 就是把一个函数体赋值给 f, f 因此可以看作函数类

型. 但是考虑到函数定义的特殊性, 因此在系统设计时我们还是区别对待, 而把这种形式看作特殊的函数定义语句.

2. 局部变量. 在函数里可以把变量声明为局部的. 局部变量不能在函数外的程序里使用, 并且在函数运行结束时会被系统消除. 即使函数外存在同名的变量, 在函数里声明的局部变量与它们也没有关系. 尽可能采用局部变量, 是对函数外部隐藏内部信息的很好方法, 这种做法使得在函数外面对于同名变量的使用不会影响本函数的行为.

在 MMP 里, 函数的局部变量来源于三个方面：参数列表, 局部变量声明 (`local`) 声明的变量, 以及在函数里未予声明的变量. 当函数里出现了一个未知的变量时, 应该怎么处理它呢？很多时候, 出现这种情况只是因为人们想使用一个临时变量, 因此把它默认当成局部变量而非全局变量是合理的.

3. 全局变量. 全局变量具有全局的作用域, 任何函数都可以通过它们的名字使用它们. MMP 规定, 函数体中对全局变量的任何使用都必须在函数开始处声明, 也就是说, 必须在关键字 global 引导的变量声明表里列出函数里需要的全局变量的名字. 完全可能发生这种情况：global 声明的全局变量在函数外部全局环境中并未找到定义, 如：

```
>f := proc()
 global x;
 x := 1;
 end proc;
```

假设在此之前 x 并未定义. 对于这种情况, MMP 系统的处理方式是在外部环境中插入 x 的变量定义, 在此之后, 声明了变量 x 的函数中对 x 的修改就意味着对全局变量 x 的修改. 因此, 在有了上面的定义之后, 如果调用 f 之后再输出 x 的值, 就会得到值 1.

4. 变量搜寻. 在函数被调用执行时, 如果在函数体里遇到变量的使用, MMP 将按照以下顺序依次搜寻, 以确定该变量的定义：

1. 函数的参数列表.
2. 全局局部的变量声明.
3. 默认为局部变量声明.

5. 递归. 允许程序员使用递归的方式定义函数. 递归常常能使程序变得更紧凑和容易理解. 但另一方面, 大量的递归也可能导致计算机资源的极大消耗, 可能占用大量存储资源, 也可能影响程序的时间性能. 递归计算的经典例子是 Fibonacci

数列：

```
>fib := proc(n)
    if n<=1 then
        return 1;
    else
        return fib(n-1 ) + fib(n-2);
    end if;
 end proc;
>fib(20);
10946
```

6. 有关函数定义的总结. 这里对函数定义的有关问题做一个总结, 供读者参考. 其中的有些内容在上面已经讨论过, 列在这里是为了帮助读者加深认识, 在编写函数时少出现相关的错误.

- 函数名以字母、下划线开头, 且由字母、数字、下划线组成. 自定义函数的名字不能与系统函数名冲突.
- 在全局变量声明中, 如果声明的是外部已经定义的变量, 则函数内部使用的就是函数调用时该全局变量当时的值. 如果外部没有定义, 则在全局环境增加这个变量的定义.
- 局部变量声明中列出的变量仅在函数内部有效. 如果出现函数中没有声明的变量, 那么都默认为局部变量.
- 全局变量声明与局部变量声明的前后位置没有关系, 但都只能出现一次. 函数的形参、全局变量声明, 以及局部变量声明三者中的变量名不能重名.
- 参数序列可以为空.
- return 语句只能出现在自定义函数的函数体中.

§3.5 MMP 编程实例

在前面几节里介绍 MMP 的编程功能时, 给出了一些非常简单的实例, 诸如 $s := 1$ 之类的小程序. 这些例子只是为了帮助读者了解 MMP 中相关结构的形式和功能. 在这一节里, 我们准备用 MMP 解决几个实际中的例子, 以帮助读者进一步理解 MMP 的编程功能, 了解一些基本的编程技术. 本节给出的编程实例围绕着一个问题：常微分方程的求解.

众所周知, 对于一般的常微分方程, 不一定存在多项式形式的解. 现在假定我

们想知道一个微分方程有没有多项式解. 若有, 则希望能得到这个解; 若无, 则希望程序报告没有多项式解. 在下面的例子里面, 我们打算用 MMP 编制一个求常微分方程的多项式解的程序, 解决这一问题. 相关算法请见 [29].

```
#####################################################################
#  Input: F        -first order autonomous differential polynomial
#         tdeg     -the total degree of F
#  Output:'easy'   -if F=y_1 or F=y_1+a
#         'yes'    -F=0 may have a polynomial general solution
#         'no'     -F=0 has no polynomial general solutiion
#####################################################################
Judgedegree:=proc(F, tdeg)
    local dy0, dy1, c, i, minu;
    dy0:=degree(F, y[0]);
    dy1:=degree(F, y[1]);
    if dy1=tdeg and dy0=dy1-1
         then if dy1=1 then return(easy)
                 else
                   minu:=F-coeff(F, y[1], dy1)*y[1]^dy1;
                # check whether the total degrees
                # of monomials in F except y_1^tdeg
                # are not greater than tdeg-1
                   for i from 0 to tdeg-1 do
                       c:=coeff(minu, y[0], i);
                       if degree(c, y[1])>tdeg-1-i
                           then return(no)
                       fi;
                   od;
                   return(yes);
              fi;
    fi;
    return(no)
end:
```

由于那些具有多项式通解的微分多项式, 它关于 y[1] 以及 y[0] 的次数必定满足一定的关系, 这里的 y[0] 代表 y 本身, y[1] 代表 y 关于 x 的导数. 上面定义的子程序主

要判断一个微分多项式的次数是否满足这一特定的关系. 在这个程序里用到了诸如 degree、coeff 之类的系统函数, 这些函数都是 MMP 提供的多项式函数, 在第二章已经介绍过. 用户可以在界面直接调用它们, 也完全可以在程序里调用它们. 这个函数的返回值很有意思, 由于程序中的 easy、yes、no 均未定义, 因此它们被 MMP 看作未定义变量. 用户可以采用下面方式判断返回值:

```
>if Judgedegree(y[1]^2-4*y[0], 2)='yes' then p:=1 end if;
p:=1
```

'yes' 表示未定义的变量 yes.

下面定义一个子程序, 用于判断求得的多项式是否为微分方程 F=0 的解.

```
######################################################################
#  Judge whether ybar is a polynomial solution of F=0
#  Input: F             -differential polynomial
#         tdeg          -total degree of F
#         ybar          -a polynomial in x
#  Output:'yes'         -ybar is a polynomial solution of F=0
#         'no'          -ybar is not polynomial solution of F=0
######################################################################
JudgeSols:=proc(F, tdeg, ybar)
    local num, i, py0, py1, ybar1;
    ybar1:=diff(ybar, x);
  # Substitute x by intergers 0. . . tdeg*(tdeg-1)/2 in F(ybar)
  # to decite whether F(ybar)=0
    for i from 0 to tdeg*(tdeg-1)/2 do
            py0:=subs(ybar, x, i);
            py1:=subs(ybar1, x, i);
            num:=subs(subs(F, y[0], py0), y[1], py1);
            if num<>0 then return(no) fi;
    od;
 # Substitute x by intergers -1. . . -tdeg*(tdeg-1)/2 in F(ybar)
 # to decite whether F(ybar)=0
    for i from -1 by -1 to -tdeg*(tdeg-1)/2 do
            py0:=subs(ybar, x, i);
            py1:=subs(ybar1, x, i);
            num:=subs(subs(F, y[0], py0), y[1], py1);
```

```
            if num<>0 then return(no) fi;
    od;
    return(yes);
end:
```

下面编写另一个函数, 它把多项式 ybar 代入 F 中, 求出 F(ybar) 中关于 x 的 deg 次项的系数.

```
####################################################################
#  Compute the coefficient of x^deg in F(ybar)
#  Input: F         -differential polynomial
#         tdeg      -total degree of F
#         ybar      -a polynomial wrt x
#         deg       -non negative integer
#         pc        -coefficients of F(y) with total degree tdeg-1
# Output: result    -the coefficient of x^deg
####################################################################
Computecoeff:=proc(F, tdeg, ybar, deg, pc)
 local py0, py1, i, j, yb0, num, result;
 num:=(tdeg-1)^2+deg-1;
 py0[0]:=1;
 py1[0]:=1;
 py0[1]:=ybar;
 py1[1]:=diff(ybar, x);
 if num=0 then
   result:=subs(subs(F,y[0],subs(py0[1],x,0)),y[1],
                subs(py1[1],x,0));
   return(result);
 fi;
# Compute ybar^i for i=1. . . tdeg-1
 for i from 2 to tdeg-1 do
   py0[i]:=expand(py0[i-1]*py0[1])
 od;
# Compute (ybar')^i for i=1. . tdeg
# (ybar' ===deravite of yabr wrt x)
 for i from 2 to tdeg do
```

```
   py1[i]:=expand(py1[i-1]*py1[1])
 od;
 for i from 0 to tdeg-1 do
   py1[i]:=pc[i]*py1[i];
 od;
 result:=0;
# Compute the coefficient of x^deg in ybar^(tdeg-2)
 if deg=0 then
   yb0:=coeff(F, y[0], tdeg-2);
   result:=result+coeff(yb0, y[1], 0)
           *coeff(py0[tdeg-2], x, num);
 fi;
# Compute the coefficient of x^deg in (ybar')^tdeg
 result:=result+coeff(F, y[1], tdeg)
         *coeff(py1[tdeg], x, num);
# Compute the coefficient of x^deg in
# ybar^(tdeg-1-i)*(ybar')^i
 if deg=0 then
  for i from 0 to tdeg-1 do
    for j from num-(tdeg-1)*i to num do
       result:=result+coeff(py0[tdeg-1-i], x, j)
               *coeff(py1[i], x, num-j);
    od;
  od;
 else
  for i from 0 to tdeg-deg do
    for j from num-(tdeg-1)*i to num do
       result:=result+coeff(py0[tdeg-1-i], x, j)
               *coeff(py1[i], x, num-j);
    od;
  od;
 fi;
 return(result);
end:
```

Computecoeff 有五个参数, 对于最后一个参数 pc, 函数定义里用到了 pc[i], 这表明 pc 是一个 table 类型或者数组类型的变量.

下面函数用于求出一个多项式, 它可能是微分方程 F=0 的解.

```
####################################################################
#  Compute a polynomial wrt x which maybe a solution of F=0
#  Input: F      -differential polynomial
#         tdeg   -total degree of F
#  Output:yb     -a polynomial wrt x
####################################################################
ComputeSols:=proc(F, tdeg)
       local coef, n, yb, i, fy, coefx, cy0, yb0, pc, deno;
       n:=tdeg;
       cy0:=coeff(F, y[0], (n-1));
       coef[n]:=-cy0/(n^n*coeff(F, y[1], n));
       coef[n-1]:=0;
       deno:=cy0*coef[n]^(n-2);
       yb:=coef[n]*x^n;
 # Compute the coefficients of monomials with total degree tdeg-1
       for i from 0 to n-1 do
         yb0:=coeff(F, y[1], i);
         pc[i]:=coeff(yb0, y[0], n-1-i);
       od;
       for i from 2 to n do
              coefx:=Computecoeff(F, tdeg, yb, n-i, pc);
              coef[n-i]:=-coefx/(deno*(i-1));
              yb:=yb+coef[n-i]*x^(n-i);
        od;
        return(expand(yb))
end:
```

有了上面的几个函数之后, 我们就可以把它们组合起来, 解决本节开始处提出的问题：求一个常微分方程的多项式解. 函数 polysolve 的输入参数是一个一阶的常系数微分多项式 F. 如果 F=0 存在多项式通解, 那么 polysolve 就会输出这一通解; 否则它给出信息, 说明 F=0 没有多项式通解或者是输入中有错.

```
polysolve:=proc(F)
```

```
      local n, sols, Gsols, a;
      n:=union_deg(F, [y[1], y[0]]);
    # the case  F is not a differential polynomial wrt y
      if n=0 then print("The Input is WRONG!");return; fi;
    # the case  F=y_1 or y_1+a
      if n>0 and Judgedegree(F, n)='easy'
            then   a:=subs(F, y[1], 0);
                   print("The Polynomial General Solutions is");
                   if a=0
                      then print(c);
                      else print(-a*(x+c));
                   fi;
                   print("The run time is");
                   return;
      fi;
   # other cases
      if n>0 and Judgedegree(F, n)='yes'
             then   sols:=ComputeSols(F, n);
                    if JudgeSols(F, n, sols)='yes' then
                                 Gsols:=subs(sols, x, x+c);
                      print("The Polynomial General Solutions is");
                                 print(Gsols);
                                 print("The run time is");
                           else
                      print("No Polynomial General Solutions");
                                 print("The run time is");
                     fi
              else
                    print("No Polynomial General Solutions");
                    print("The run time is");
      fi;
end:
```

函数 polysolve 里采用的主要算法概述如下：

1. 求 F 作为多项式时的全次数 n. 如果 n=0 则说明输入有错, 程序终止.

2. 调用子程序 Judgedegree, 判断 F 关于 y[1] 以及 y[0] 的次数是否满足特定的关系. 如果 F 不满足该条件, 输出：F=0 无多项式通解. 如果 F 是满足条件的平凡的微分多项式, 则直接求解并输出结果, 程序终止.
3. 调用子程序 ComputeSols, 求出一个多项式 yb . 如果 F=0 存在多项式解, 则 yb 必定是 F=0 的解.
4. 调用子程序 JudgeSols , 判断第 3 步中所求 yb 是否是 F=0 的解.

有了 polysolve 函数, 就可以求解一阶常微分方程的多项式解.

```
>polysolve(y[1]^2-4*y[0]);
"The Polynomial General Solutions is"
c^2+2*x*c+x^2
>polysolve(y[1]-4*y[0]);
"No Polynomial General Solutions"
```

第一个例子说明微分方程 $y'^2-4y=0$ 的通解是 $y=(x+c)^2$, 其中 c 是任意常数. 第二个例子说明微分方程 $y'-4y=0$ 的通解不是多项式形式.

上面例子显示了利用 MMP 一步一步地解决一个较复杂问题的过程. 看过这个例子, 读者应该对用 MMP 编程有了一定的了解.

第四章　吴特征列方法

吴特征列方法是由吴文俊院士提出的求解多项式方程组的一种一般方法, 该方法以多项式方程组的零点集为基本关注点, 给出了多项式组零点集的构造性描述, 并在此基础之上给出了多项式方程组解的结构. 吴特征列方法是数学机械化平台 MMP 的核心算法, 同时也是 MMP 中几何自动推理、代数方程求解、微分方程求解以及其他应用模块的基础. 本章主要介绍代数、常微、偏微情形的吴特征列方法与投影算法.

§4.1　多项式与升列

本节将介绍特征列方法的核心概念：升列. 为节省篇幅, 这里只对最要紧的概念和定理给出简要的介绍. 更深入的探讨请参阅 [86] 和 [87].

1. 多项式的伪余式. 设 $\mathbb{K}$ 是特征为零的数域, $\mathbb{K}[x_1,\cdots,x_n]$ 是基域为 $\mathbb{K}$, 变元为 $x_1,\cdots,x_n$ 的多项式环, 简记作 $\mathbb{K}[\mathbb{X}]$, 这里 $\mathbb{X}=\{x_1,\cdots,x_n\}$. 除非另作说明, 本章中出现的多项式和多项式集都是指在 $\mathbb{K}[\mathbb{X}]$ 中. 对于变元的序我们总是假定有序 $x_1<\cdots<x_n$.

对一个非零多项式 $P\in\mathbb{K}[\mathbb{X}]$ 与给定的变元 x_i, P 对 x_i 的*次数*, 记为 $\deg(P,x_i)$. 如果这一次数非 0, 那么 P 对 x_i 的*首项系数*, 记为 $\mathrm{lc}(P,x_i)$. P 的类定义为 P 中出现的 x_c 的最大下标 c; 如果 P 是 $\mathbb{K}$ 中一个非零常量, 则 P 的类定义为 0. P 的类记作 $\mathrm{cls}(P)$. 类为 c 的多项式 P 可以写作

$$P=C_0x_c^d+C_1x_c^{d-1}+\cdots+C_d$$

其中 $C_0,\cdots,C_d$ 是 $\mathbb{K}[x_1,\cdots,x_{c-1}]$ 中的多项式, 且 $d>0$, $C_0\neq 0$. x_c 称作 P 的主变元, 记为 $\mathrm{lvar}(P)$, C_0 称为 P 的 *初式*, 记为 $\mathrm{I}(P)$. 将 $C_0x_c^d$ 称为 P 的 *首项*, 记为 $\mathrm{lterm}(P)$, 而 $C_1x_c^{d-1}+\cdots+C_d$ 称为 P 的 *余项*, 记为 $\mathrm{red}(P)$. $\frac{\partial P}{\partial x_c}$ 定义为 P 的 *隔离子*, 记为 $\mathrm{S}(P)$.

在 MMP 中, 多项式集合用多项式的链表表示, 比如 $F:=[x^2+5-2*x*z, z^3*y+x*y^2, -8*z^3+3*y^2]$. 在本章介绍的例子中, 我们总是用 F 代表上面给出的多项式组.

在 MMP 中, 我们也用表结构表示变元的序. 比如一个表 $[x, y, z]$ 用来表示一个变元的序时, 则意味着变元的序为 $x > y > z$. 我们来看下面的例子. MMP 函数 mainvar 用来计算一个多项式在一个给定序下的主变元.

```
>p1:=x^2+5-2*x*z:
>p2:=z^3*y+x*y^2:
>p3:=-8*z^3+3*y^2:
>F:=[p1, p2, p3];
[x^2+5-2*x*z, z^3*y+x*y^2, -8*z^3+3*y^2]
>mainvar(p1, [x, y, z]);
x
>mainvar(p1, [z, y, x]);
z
```

设 P, Q 为 $\mathbb{K}[\mathbb{X}]$ 中的非零多项式, 称 P 的序低于Q 的序, 记作 $P \prec Q$: 如果 $\mathrm{cls}(P) < \mathrm{cls}(Q)$, 或 $\mathrm{cls}(P) = \mathrm{cls}(Q) = c$ 但 $\deg(P, x_c) < \deg(Q, x_c)$. 如果 $P \prec Q$ 和 $Q \prec P$ 都不成立, 称 P 和 Q 有相同的序, 记作 $P \sim Q$. 这样我们对 $\mathbb{K}[\mathbb{X}]$ 中的多项式建立了一个全序. 如果 $\mathrm{cls}(Q) = c > 0$ 且 $\deg(P, x_c) < \deg(Q, x_c)$. 则称 P 关于 Q 是约化的.

设 F, G 为非零多项式, 算法 prem 用来计算多项式 F 对多项式 G 关于变元 x 的伪余式. 为方便起见, 伪余式也称余式.

Algorithm 1: $\mathrm{prem}(F, G, x)$

Input: 多项式 F, G, 变元 x.

Output: F 对 G 的关于变元 x 的伪余式.

$R = F$

if $\deg(G, x)$=0 **then return** 0

if $\deg(F, x)$<$\deg(G, x)$ **then return** R

lc_g =$\mathrm{lc}(G, x)$

d_g =$\deg(G, x)$

while $\deg(R, x) \geqslant \deg(G, x)$ **do**

  lc_r =$\mathrm{lc}(R, x)$

  d_r =$\deg(R, x)$

  $R = lc_g * R - lc_r * G * x^{(d_r - d_g)}$

end

return R

设 F 对 G 关于变元 x 的余式为 R, 我们有下面的 *余式公式*

$$\mathrm{lc}(G,x)^s F = QG + R \tag{4.1}$$

其中 s 为非负整数, 而且 $\deg(R,x) < \deg(G,x)$. 如果 x 是 G 的主变元, 那么 R 关于 G 是约化的. 当 s 尽可能小时, 上式中的余式 R 是唯一的.

MMP 函数 prem 用来计算两个多项式对指定变元的余式. 继续前面的例子:

```
>prem(p1, p2, x);
y*z^6+2*y^2*z^4+5*y^3
>prem(p2, p1, z);
-y*x^6-8*y^2*x^4-15*y*x^4-75*y*x^2-125*y
>prem(p3, p2, y);
-8*z^3*x-3*y*z^3
```

2. 升列. 一个多项式序列 $\mathcal{A}$: $A_1, A_2, \cdots, A_r$ 称作是一个*三角列*, 如果 $r=1$ 且 $A_1 \neq 0$, 或者 $\mathrm{cls}(A_1) < \mathrm{cls}(A_2) < \cdots < \mathrm{cls}(A_r)$. 如果三角列 $\mathcal{A}$ 进一步满足条件: 对任意的 $i<j$, A_j 关于 A_i 是约化的, 则称作是一个*升列*. 如果 $r=1$, 而且 A_1 是 K 中的非 0 常数, 则称 $\mathcal{A}$ 是一个*矛盾列*. 上面定义的升列也称为 Ritt 升列

对于升列 $\mathcal{A}$: $A_1, A_2, \cdots, A_r$, 我们用 $\mathcal{A}_i, i=1,\cdots,r$ 表示升列 $A_1,\cdots,A_i$, 用 $\mathrm{I}_{\mathcal{A}}$ 表示 $\mathcal{A}$ 中多项式初式的乘积. 升列 $\mathcal{A}$: $A_1, A_2, \cdots, A_r$ 称作是*拟线性*的, 如果对 $i>1$, $\deg(A_i, x_{\mathrm{cls}(A_i)}) = 1$.

设 $\mathcal{A}$: $A_1,\cdots,A_r$ 与 $\mathcal{B}$: $B_1,\cdots,A_s$ 是两个升列, 如果以下的 (1) 或 (2) 中之一成立, 则称 $\mathcal{A}$ 的序*低于* $\mathcal{B}$ 的序, 记作 $\mathcal{A} \prec \mathcal{B}$.

(1) 存在 $1 \leqslant k \leqslant \min(r,s)$, 使得 $A_1 \sim B_1, \cdots, A_{k-1} \sim B_{k-1}$, 但 $A_k \prec B_k$.

(2) $r > s$, 且 $A_1 \sim B_1, \cdots, A_s \sim B_s$.

如果 $\mathcal{A} \prec \mathcal{B}$ 和 $\mathcal{B} \prec \mathcal{A}$ 都不成立, 称升列 $\mathcal{A}$,$\mathcal{B}$ 有相等的序, 记作 $\mathcal{A} \sim \mathcal{B}$.

一个多项式 P 称作是关于升列 $\mathcal{A}$*约化*的, 如果 P 关于 $\mathcal{A}$ 中的所有多项式都是约化的.

设 $\mathcal{A}$: $A_1, A_2, \cdots, A_r$ 是一个升列, P 是一个多项式, 我们可以按下面的形式定义多项式 P 关于升列 $\mathcal{A}$ 的余式. 首先令 $R_r = P$. 设 R_{i-1} 为 R_i 对 A_i 的余式, $i=r,\cdots,1$, R_0 就称为 P 对升列 $\mathcal{A}$ 的余式. 而且该余式对升列 $\mathcal{A}$ 是约化的. 由 (4.1), 我们有

$$\begin{gathered} I_r^{s_r} R_r = Q_r A_r + R_{r-1} \\ I_{r-1}^{s_{r-1}} R_{n-1} = Q_{r-1} A_{r-1} + R_{r-2} \\ \cdots\cdots\cdots\cdots \\ I_1^{s_1} R_1 = Q_1 A_1 + R_0 \end{gathered}$$

设 $R = R_0$, 从上式可以得到

$$I_1^{s_1} I_2^{s_2} \cdots I_r^{s_r} P = Q_1' A_1 + \cdots + Q_r' A_r + R \tag{4.2}$$

这儿 Q_i' 是多项式, $I_i = \mathrm{I}(A_i)$, s_i 是非负整数, 我们称上式为*余式公式*. 算法 premas 用来计算多项式对升列的余式.

Algorithm 2: premas($P, \mathcal{A}$)

Input: 多项式 P, 升列 $\mathcal{A} : A_1, A_2 \cdots, A_r$
Output: R, 为 P 对升列 AS 的余式
$R = P$
$i = r$
while $i \neq 0$ **do**
  $x =$lvar(A_i)
  $R =$prem(R, A_i, x)
  $i = i - 1$
end
return R

MMP 的函数 premas 计算多项式对升列的余式.

```
>B:=[x^2-2*z*x+5, -8*z^3+3*y^2]:
>p2:=y^2*x+y*z^3:
>premas(p2, B, [x, y, z]);
8*z^3*x+3*y*z^3
```

多项式集合 $\mathbb{P}$ 中的多项式关于升列 $\mathcal{A}$ 的所有非零余式组成的集合称作 $\mathbb{P}$ 关于 $\mathcal{A}$ 的*余式集*. 记作 remset$(\mathbb{P}, \mathcal{A})$. 多项式集合 $\mathbb{P}$ 对于 $\mathcal{A}$ 的余式集合 $\mathbb{R}$ 关于 $\mathcal{A}$ 是约化的. 算法 remset 计算多项式集合对升列求余的余式的集合.

Algorithm 3: remset($\mathbb{P}, \mathcal{A}$)

Input: 多项式集合 $\mathbb{P}$, 升列 $\mathcal{A}$
Output: $\mathbb{R}$, 为 $\mathbb{P}$ 对升列 $\mathcal{A}$ 的余式集合
$\mathbb{R} = \emptyset$
foreach $p \in \mathbb{P}$ **do**
  $r =$premas$(p, \mathcal{A})$
  if $r \neq 0$ **then** $\mathbb{R}$=$\mathbb{R} \cup \{r\}$
end
return $\mathbb{R}$

MMP 中的函数 remset 用来计算多项式集合对升列的余式.

```
>A:=[x^2-2*z*x+5]:
>B:=[x^2-2*z*x+5, -8*z^3+3*y^2]:
>remset(F, A, [z, y, x]);
[-y*x^6-8*y^2*x^4-15*y*x^4-75*y*x^2-125*y,
 4*x^6+60*x^4-12*y^2*x^3+300*x^2+500]
>remset(F, B, [x, y, z]);
[8*z^3*x+3*y*z^3]
```

设 $\mathbb{P}$ 是一个多项式集合, 我们用 $(\mathbb{P})$ 表示由 $\mathbb{P}$ 中的多项式在 $\mathbb{K}[\mathbb{X}]$ 中生成的理想. 设 $\mathcal{A}$ 是一个升列, 我们定义集合：

$$\mathrm{sat}(\mathcal{A}) = \{P|P \in \mathbb{K}[\mathbb{X}], \text{存在非负整数}s, \text{使得}\mathrm{I}_{\mathcal{A}}^s P \in (\mathcal{A})\}$$

容易证明 $\mathrm{sat}(\mathcal{A})$ 是 $\mathbb{K}[\mathbb{X}]$ 中的一个理想, 我们称它为升列 $\mathcal{A}$ 的*饱和理想*. 我们有下面的结论 ([35])：

引理 4.1.1 设 $\mathcal{A}$ 是一个升列, 则 $\mathrm{sat}(\mathcal{A}) = (\mathcal{A}, \mathrm{I}_{\mathcal{A}}z - 1) \cap \mathbb{K}[\mathbb{X}]$, 其中 z 是一个新变量.

利用上述引理与 Gröbner 基算法, 可以求得 $\mathrm{sat}(\mathcal{A})$ 的一组基. 有关 Gröbner 基的一些基本性质请参考 [9].

用 $\mathrm{Res}(P, Q, x_r)$ 表示多项式 P, Q 关于变元 x_r 的结式. 设 $\mathcal{A} : A_1, A_2, \cdots, A_r$ 是一个升列, A_i 的主变元为 y_i. P 是任意一个多项式, 令 $R_r = P$, $R_{r-1} = \mathrm{Res}(R_r, A_r, y_r), \cdots, R_0 = \mathrm{Res}(R_1, A_1, y_1)$, R_0 称为多项式 P 对一个升列 $\mathcal{A}$ 的结式, 记为 $\mathrm{Res}(P, \mathcal{A})$.

§4.2 整序原理

特征列方法的目的是给出多项式方程组零点集的构造性描述. 为此, 我们先定义零点集以及与之相关的代数簇.

1. 代数簇与拟代数簇. $\mathbb{E}$ 为包含 $\mathbb{K}$ 的一个泛扩域, 即在 $\mathbb{K}$ 上添加无穷多个未定元后所得扩域的代数闭域. $\mathbb{E}^n$ 是 $\mathbb{E}$ 上的 n 维仿射空间. 对于 $\mathbb{K}[x_1, \cdots, x_n]$ 上的一个多项式 F, $\mathbb{E}^n$ 中的点 $\xi = (\xi_1, \cdots, \xi_n)$ 称为 F 的一个零点, 如果 $F(\xi_1, \cdots, \xi_n) = 0$. 设 $\mathbb{P} = \{P_1, \cdots, P_s\}$ 为 $\mathbb{K}[\mathbb{X}]$ 中的多项式集合, $\mathbb{E}^n$ 中的点 $\xi = (\xi_1, \cdots, \xi_n)$ 称为 $\mathbb{P}$ 的一个零点, 如果 $P_i(\xi) = 0, i = 1, \cdots, s$. $\mathbb{P}$ 的所有零点组成的集合称作 $\mathbb{P}$ 的*零点集*. 一般用 $\mathrm{Zero}(\mathbb{P})$ 表示. $\mathbb{E}^n$ 上的一个*代数簇*定义为 $\mathbb{K}[\mathbb{X}]$ 中的某个多项式集合的零点集.

设 $\mathbb{P}$ 是一个多项式集合, D 是一个多项式, 我们定义：

$$\mathrm{Zero}(\mathbb{P}/D)=\{\xi\in\mathbb{E}^n|\xi\in\mathrm{Zero}(\mathbb{P}),D(\xi)\neq 0.\}$$

设 $\mathbb{P},\mathbb{D}$ 是两个多项式集合, 我们定义

$$\mathrm{Zero}(\mathbb{P}/\mathbb{D})=\{\xi\in\mathbb{E}^n|\xi\in\mathrm{Zero}(\mathbb{P}),\text{对}\mathbb{D}\text{中任意的多项式}D,\text{都有}D(\xi)\neq 0.\}$$

设 $\mathbb{P}_i,\mathbb{D}_i$, $i=1,\cdots,m$ 是多项式集合, 我们称集合 $\cup_{i=1}^m\mathrm{Zero}(\mathbb{P}_i/\mathbb{D}_i)$ 为一个*拟代数簇*.

代数簇 $V_1=\mathrm{Zero}(\mathbb{P}_1)$ 称作代数簇 $V_2=\mathrm{Zero}(\mathbb{P}_2)$ 的*子簇*, 如果 $V_1\subseteq V_2$. 称 V_1 是 V_2 的*真子簇*, 如果 $V_1\subset V_2$ 且 $V_1\neq V_2$.

代数簇 $V=\mathrm{Zero}(\mathbb{P})$ 称作在 $\mathbb{K}$ 上*不可约*, 如果 V 不能写成两个真子簇的并. V 称作是可约的, 如果 V 不是不可约的.

一个多项式集 $\mathbb{P}$ 的*零点结构*是指 $\mathbb{P}$ 的零点集的结构, 即代数簇 $\mathrm{Zero}(\mathbb{P})$ 的结构. 下面对于一个给定的多项式集 $\mathbb{P}$, 我们通过其特征列研究它的零点结构. 关于代数簇的一个基本结果是 ([86]):

定理 4.2.1 *任意一个代数簇可以分解为互不包含的不可约代数簇的并.*

本章将要介绍的特征列方法将给出上述定理一个构造性证明.

2. 升列的零点. $\mathcal{A}$ 是一个升列, 如果 $\xi\in\mathrm{Zero}(\mathcal{A})$, 而且 $\mathrm{I}_{\mathcal{A}}(x)\neq 0$, 则称 ξ 称为升列 $\mathcal{A}$ 的一个*正常零点*. $\mathcal{A}$ 的所有正常零点的集合是 $\mathrm{Zero}(\mathcal{A}/\mathrm{I}_{\mathcal{A}})$.

P 是一个多项式, $\mathcal{A}$ 是一个升列, 如果 P 对 $\mathcal{A}$ 的余式为 0, 我们有

$$\mathrm{Zero}(\mathcal{A}/\mathrm{I}_{\mathcal{A}})\subset\mathrm{Zero}(P)$$

设 $\mathcal{A}:A_1,\cdots,A_p$ 是一个升列. 可以将 $x_1,\cdots,x_n$ 重新命名为 $u_1,\cdots,u_q$, $y_1,\cdots,y_p,(p+q=n)$, 使得 $\mathrm{lvar}(A_i)=y_i$. 我们可以将 $\mathcal{A}$ 写为

$$\begin{array}{lcl}A_1(\mathbb{U},y_1) & = & I_1*y_1^{d_1}\ +\text{关于 } y_1 \text{ 的低次项}\\ A_2(\mathbb{U},y_1,y_2) & = & I_2*y_2^{d_2}\ +\text{关于 } y_2 \text{ 的低次项}\\ & & \cdots\cdots\cdots\cdots\\ A_p(\mathbb{U},y_1,\cdots,y_p) & = & I_p*y_p^{d_p}\ +\text{关于 } y_p \text{ 的低次项}\end{array}\tag{4.3}$$

变量 $\mathbb{U}=\{u_1,\cdots,u_q\}$ 称为 $\mathcal{A}$ 的*参变量*. 参变量的个数称为 $\mathcal{A}$ 的*维数*, 记为 $\dim(\mathrm{A})$. 一个升列的解空间是基本确定的. 例如, 给定 u_i 的一组值, 如果所有的初式 I_i 都不为零, 我们一般可以由 $A_1=0,A_2=0,\cdots,A_p=0$ 顺序求解 $y_1,y_2,\cdots,y_p$. 而且解的个数一般是 $\prod_{i=1}^p d_i$. 换言之, $\mathrm{Zero}(\mathcal{A}/\mathrm{I}_{\mathcal{A}})$ 可以被认为是一个可确定的集合. 一般地, 我们有下面的结果 ([35])：

定理 4.2.2 设 $\mathcal{A}$ 是升列, J 是 $\mathcal{A}$ 的初式的乘积. 则有 $\mathrm{Zero}(\mathcal{A}/J)$ 与 $\mathrm{Zero}(\mathrm{sat}(\mathcal{A}))$ 或者是空集或者是齐维的, 且其分支的维数是 $\dim(\mathcal{A})$.

例 4.2.3 对于升列 $\mathcal{A}$, $\mathrm{Zero}(\mathcal{A})$ 不具有任何几何意义. 设 $\mathcal{A} = x_1^2, x_1x_2, x_1x_3$. 则 $\mathrm{Zero}(\mathcal{A}) = \mathrm{Zero}(x_1)$ 是二维的. 设 $\mathcal{A} = x_1^2, x_1x_2, x_3$. 则 $\mathrm{Zero}(\mathcal{A}) = \mathrm{Zero}(x_1, x_3)$ 是一维的. 设 $\mathcal{A} = x_1^2, x_1x_2 - 1$. 则 $\mathrm{Zero}(\mathcal{A}) = \emptyset$. 此时定义其维数为 -1.

3. 基列. 对于升列的序, 我们有以下的结论.

引理 4.2.4 序严格降低的升列序列是有限的.

由上述引理, 一个多项式集合 $\mathbb{P}$ 中的多项式所组成的所有升列中存在一个序最小的升列, 这一升列称为 $\mathbb{P}$ 的基列. 容易证明理想中的任意一个多项式对它的基列求余的余式都是零. 基列满足下面性质.

引理 4.2.5 在一个多项式集合中加入一个对于其基列约化的多项式, 则新的多项式集合的基列的序比原集合的基列的序要低.

算法 basicset 用来计算一个多项式集合的基列.

Algorithm 4: basicset($\mathbb{P}$)

Input: 多项式集合 $\mathbb{P}$
Output: $\mathcal{B}$, 为 $\mathbb{P}$ 的基列.
$\mathcal{B} \neq \emptyset$
while $\mathbb{P}$ 非空 **do**
  b=$\mathbb{P}$ 中秩最低的多项式
  $\mathcal{B} = \mathcal{B} \cup \{b\}$
  $\mathcal{P}' = \emptyset$
  foreach p *in* $\mathbb{P}$ **do**
   if p 对 $\mathcal{B}$ 约化 **then** $\mathcal{P}' = \mathcal{P}' \cup \{p\}$
  end
  $\mathbb{P} = \mathcal{P}'$
end
return $\mathcal{B}$

MMP 中的函数 basicset 用来计算给定的多项式组在给定序下的基列. 其中的 F 在第 80 页给出.

```
>basicset(F, [z, y, x]);
[x^2-2*z*x+5]
>basicset(F, [x, y, z]);
[x^2-2*z*x+5, -8*z^3+3*y^2]
```

4. 特征列. 设 $\mathbb{P}$ 是一个多项式集合. 我们将给出计算其特征列的算法. 先设 $\mathbb{P}_0 = \mathbb{P}$, 再任选 $\mathbb{P}_0$ 的一个基列 $\mathcal{B}_0$. 如果存在多项式属于集合 $\mathbb{P}_0$, 但不属于 $\mathcal{B}_0$, 那么就求出这些多项式关于 $\mathcal{B}_0$ 的余式, 非零的余式集合记为 $\mathbb{R}_0$, 于是我们可得到一个新的多项式集 $\mathbb{P}_1 = \mathbb{P}_0 \cup \mathbb{R}_0$. 现在再计算 $\mathbb{P}_1$ 的基列 $\mathcal{B}_1$. 由于 $\mathbb{R}_0$ 中的多项式对 $\mathcal{B}_0$ 是约化的. 根据引理 4.2.5, $\mathcal{B}_1$ 的序要比 $\mathcal{B}_0$ 的序低. 如果 $\mathbb{P}_1$ 关于 $\mathcal{B}_1$ 的余式里存在非零多项式, 我们就又得到一个余式集 $\mathbb{R}_1$. 假设 $\mathbb{R}_1$ 非空, 那么我们形成多项式集合 $\mathbb{P}_2 = \mathbb{P}_1 \cup \mathbb{R}_1$, 以及和前面类似的基列 $\mathcal{B}_2$ 和余式集 $\mathbb{R}_2$. 用这种方法不断做下去, 就得到一系列的非空余式集, 如下所示：

$$\mathcal{B}_0 \succ \mathcal{B}_1 \succ \mathcal{B}_2 \succ \cdots$$

根据引理 4.2.4, 这个升列序列一定有限, 因此一定存在 m, 有 $\mathbb{R}_m = \emptyset$. 最后的基列 $\mathcal{B}_m$ 记为 $\mathcal{C}$. 以上过程可以表示如下：

$$\begin{array}{lllllll} \mathbb{P} = & \mathbb{P}_0 & \mathbb{P}_1 & \cdots & \mathbb{P}_i & \cdots & \mathbb{P}_m \\ & \mathcal{B}_0 & \mathcal{B}_1 & \cdots & \mathcal{B}_i & \cdots & \mathcal{B}_m = \mathcal{C} \\ & \mathbb{R}_0 & \mathbb{R}_1 & \cdots & \mathbb{R}_i & \cdots & \mathbb{R}_m = \emptyset \end{array} \tag{4.4}$$

在上面的过程中, 对 $i < m$, 余式集 $\mathbb{R}_i$ 非空. 我们有

$$\begin{aligned} &\mathcal{B}_i = \mathbb{P}_i \text{的一个基列} \\ &\mathbb{R}_i = \mathrm{remset}(\mathbb{P}_i, \mathcal{B}_i) \\ &\mathbb{P}_i = \mathbb{P}_{i-1} \cup \mathbb{R}_{i-1} \end{aligned}$$

引理 4.2.6　(4.4)所表示的过程在有限的 m 步之后终止, 此时相应的余式集 $\mathbb{R}_m = \emptyset$. 相应的基列 $\mathcal{B}_m = \mathcal{C}$ 满足下列性质：

$$\mathrm{remset}(\mathbb{P}, \mathcal{C}) = \emptyset \text{ 且} \mathrm{Zero}(\mathbb{P}) \subset \mathrm{Zero}(\mathcal{C}) \tag{4.5}$$

一个多项式集合 $\mathbb{P}$ 的特征列是指满足 (4.5) 的任一升列 $\mathcal{C}$. 我们有

定理 4.2.7 (整序原理)　设 $\mathcal{C} : C_1, \cdots, C_r$ 为多项式集合 $\mathbb{P}$ 的特征列. 那么

$$\begin{aligned} &\mathrm{Zero}(\mathcal{C}/I_{\mathcal{C}}) \subset \mathrm{Zero}(\mathbb{P}) \subset \mathrm{Zero}(\mathcal{C}) \\ &\mathrm{Zero}(\mathbb{P}/I_{\mathcal{C}}) = \mathrm{Zero}(\mathcal{C}/I_{\mathcal{C}}) \\ &\mathrm{Zero}(\mathbb{P}) = \mathrm{Zero}(\mathcal{C}/I_{\mathcal{C}}) \bigcup \cup_{i=1}^{r} \mathrm{Zero}(\mathbb{P} \cup \{\mathrm{I}(C_i)\}) \\ &\mathrm{Zero}(\mathbb{P}) = \mathrm{Zero}(\mathrm{sat}(\mathcal{C})) \bigcup \cup_{i=1}^{r} \mathrm{Zero}(\mathbb{P} \cup \{\mathrm{I}(C_i)\}) \end{aligned}$$

一个多项式集合 $\mathbb{P}$ 的特征列可以按照过程 (4.4) 中的步骤由 $\mathbb{P}$ 求得. 若特征列 $\mathcal{C}$ 是平凡的, 即只含有一个非零常量, 则 $\mathrm{Zero}(\mathbb{P}) = \emptyset$. 对于多项式集合 $\mathbb{P}$ 的一个特

征列 $\mathcal{C}$, 并不要求 $\mathcal{C}$ 中的多项式在 $\mathbb{P}$ 中. 然而, 从过程中可以看出 $\mathcal{C}$ 中的多项式都在以 $\mathbb{P}$ 的多项式为基的生成理想中. 多项式集合 $\mathbb{P}$ 的特征列 $\mathcal{C}$ 并不唯一, 这是因为在过程 (4.4) 中, 每一步对基列的选择不唯一.

算法 charset 用来计算一个多项式集合的特征列.

MMP 中的函数 charset 计算一个多项式集合的特征列, 其中的 F 在第 80 页给出.

```
>charset(F, [z, y, x]);
[x^2-2*z*x+5,
 32*x^7+12*y*x^6+480*x^5+180*y*x^4+2400*x^3+900*y*x^2+4000*x+1500*y,
 576*x^12-12288*x^11+17280*x^10-184320*x^9+216000*x^8-921600*x^7+
 1440000*x^6-1536000*x^5+5400000*x^4+10800000*x^2+9000000]
```

Algorithm 5: charset($\mathbb{P}$)

Input: 多项式集合 $\mathbb{P}$
Output: 升列 $\mathcal{C}$: 为 $\mathbb{P}$ 的特征列.
$\mathbb{R}=\mathbb{P}$
while $\mathbb{R}\neq\emptyset$ **do**
  $\mathcal{C}$ =basicset($\mathbb{P}$)
  $\mathbb{R}$ =remset($\mathbb{P}$,$\mathcal{C}$)
  $\mathbb{P}=\mathbb{P}\cup\mathbb{R}$
end
return $\mathcal{C}$

§4.3 代数情形的零点分解算法

1. 零点分解算法. 将整序原理中的第三个等式作适当的修改得到

$$\mathrm{Zero}(\mathbb{P})=\mathrm{Zero}(\mathcal{C}/I_{\mathcal{C}})\bigcup\cup_{i=1}^{r}\mathrm{Zero}(\mathbb{P}\cup\mathcal{C}\cup\{\mathrm{I}(C_i)\})$$

需要注意到 $\mathrm{I}(C_i)$ 对 $\mathcal{C}$ 已约化, 因此, $\mathbb{P}\cup\mathcal{C}\cup\mathrm{I}(C_i)$ 的基列的秩低于 $\mathcal{C}$ 的秩. 对每一个 $\mathbb{P}\cup\mathcal{C}\cup\mathrm{I}(C_i)$, 再递归调用整序原理的过程. 由引理 4.2.4, 这样的递归过程在有限步内一定终止. 因此我们得到下面的定理.

定理 4.3.1 (零点分解定理) *存在一个算法, 对于给定的多项式集 $\mathbb{P}$, 或者判定 $\mathrm{Zero}(\mathbb{P})=\emptyset$, 或者可在有限步内得到有限个升列 $\mathcal{C}_i$, 使得*

$$\mathrm{Zero}(\mathbb{P})=\cup_i\mathrm{Zero}(\mathcal{C}_i/I_{\mathcal{C}_i})$$

对于每一个 i, 有 $\operatorname{remset}(\mathbb{P},\mathcal{C}_i)=\emptyset$.

由上述零点分解定理中得到的一组升列 $\mathcal{C}_i$ 称为 $\mathbb{P}$ 的一个特征列序列. 函数 charser 用来计算一个多项式集合的特征列序列. 下例中的 F 在第 80页给出.

```
>charser(F, [z, y, x]);
[[x^2-2*z*x+5, 32*x^7+12*y*x^6+480*x^5+180*y*x^4+
  2400*x^3+900*y*x^2+4000*x+1500*y,
  576*x^12-12288*x^11+17280*x^10-184320*x^9+216000*x^8-
  921600*x^7+1440000*x^6-1536000*x^5+5400000*x^4+
  10800000*x^2+9000000],
  [x^2-2*z*x+5, -36*y^2*x^3, 12*x^6+180*x ^4+900*x^2+1500]]
```

上述命令将多项式集合 F 的零点分解为两部分.

定理 4.3.1 在理论上给出了零点分解的基本算法. 在实现这一算法时, 为了提高效率, 我们对原算法进行各种改进. MMP 系统中的一个重要函数 wsolve 用来给出各种形式的零点分解. wsolve 的调用方法如下：

```
wsolve(F, Y)
wsolve(F, Y, D)
wsolve(F, Y, D, type)
参数: F  -  变元为U,Y的多项式的表.
      Y  -  变元表, Y同时给出变元的次序. U中的变元当作系数.
      D  -  变元为Y的多项式的表, 可以是空表.
      type  -  只能取 "ritt", "wu", "weak", "regular", "normal",
               "saturated", "irreducible", "seidbg", "finite" 之一.
               缺省值为"ritt".
输出: 一个升列构成的表.
```

命令 wsolve(F, Y, D, type) 计算出 F 在变元序 Y 下的具有 ‘type’ 形式的一组特征列. 输出中的升列的正常零点一定是原始输入的多项式组的零点. 原始输入多项式的零点一定是输出中的某个升列的正常零点. 多项式集合 D 用来尽可能除去一些升列. 如果输出是 $[\mathcal{A}_1,\cdots,\mathcal{A}_r]$, 那么有

$$\operatorname{Zero}(F/D)=\cup_{i=1}^{r}\operatorname{Zero}(\mathcal{A}_i/I_{\mathcal{A}_i}D)$$

而且 $\mathcal{A}_i$ 是参数 ‘type’ 指定形式的升列. 升列的各种形式将在本节详细介绍.

2. 多项式因式分解与零点分解. 在特征列的计算过程中, 余式的计算是其中计算量最大的一个运算. 在多项式对升列的余式计算中, 一般来说, 余式集中的多项式的项数或总次数会变得越来越大. 从而使得计算特征列变得很困难. 将多项式在有

理数域上作因式分解, 是降低多项式次数和项数的一个有效的方法. 因此, 我们将尽可能的对在计算中出现的多项式在有理数域上作因式分解. wsolve 充分利用了多项式因式分解来简化零点分解算法.

设 $\mathbb{P} = \{P_1, \cdots, P_r\}$ 是一个多项式集合. 我们定义一个集合 $\mathrm{F}(\mathbb{P}) = \{\{F_1, \cdots, F_r\} | F_i \text{是} P_i \text{在有理数域上的不可约因式}, \quad i = 1, \cdots, r\}$, 将这一想法与零点分解定理结合得到零点分解的一种新形式.

Algorithm 6: scs($\mathbb{P}$)

Input: 多项式组 $\mathbb{P}$

Output: 有限个升列 $\mathcal{AS}_i$, 使得 $\cup_i \mathrm{Zero}(\mathcal{A}_i / I_{\mathcal{A}_i}) \subset \mathrm{Zero}(\mathbb{P}) \subset \cup_i \mathrm{Zero}(\mathcal{A}_i)$

$\mathcal{B} =$basicset($\mathbb{P}$)

$\mathbb{R} =$remset($\mathbb{P}, \mathcal{B}$)

if $R = \emptyset$ **then return** $\mathcal{B}$

$\mathbb{R}' =$F($\mathbb{R}$)

$\mathcal{A} = \emptyset$

foreach $R'' \in \mathbb{R}'$ **do** $\mathcal{A} = \mathcal{A} \cup \mathrm{scs}(\mathbb{P} \cup R'')$

return $\mathcal{A}$

结合算法 SCS 和特征列序列的算法, 我们可以有下面的结论:

定理 4.3.2 *设 $\mathbb{P}$ 为一个多项式组, 在有限步内可以得到有限个升列 $\mathcal{A}_i$, 使得有下列形式的零点分解*

$$\mathrm{Zero}(\mathbb{P}) = \cup_i \mathrm{Zero}(\mathcal{A}_i / I_{\mathcal{A}_i})$$

其中升列 $\mathcal{A}_i$ 中的多项式在 $\mathbb{K}[\mathbb{X}]$ 中不可约.

下面是关于 wsolve 的例子:

```
>F3:=[x3+x2+x1, x3*x2+x2*x1+x1*x3, x3*x2*x1-1];
>wsolve(F3, [x3, x2, x1]);
Time is: 0.14 second(s).
[[x3+x2+1, x2^2+x2+1, x1-1],
 [x3+x1+1, x2-1, x1^2+x1+1],
 [x3-1, x2+x1+1, x1^2+x1+1]]
```

此例即所谓 cyclic-3 问题. cyclic-4 问题已在 § 1.4 讨论. 使用 wsolve, 我们得到

$$\mathrm{Zero}(F_3) = \mathrm{Zero}(\mathcal{A}_1) \cup \mathrm{Zero}(\mathcal{A}_2) \cup \mathrm{Zero}(\mathcal{A}_3)$$

其中 $\mathcal{A}_1 = x_3 + x_2 + 1, x_2^2 + x_2 + 1, x_1 - 1$; $\mathcal{A}_2 = x_3 + x_1 + 1, x_2 - 1, x_1^2 + x_1 + 1$; $\mathcal{A}_3 = x_3 - 1, x_2 + x_1 + 1, x_1^2 + x_1 + 1$. 容易知道 $\mathcal{A}_1 = 0$ 有两组解: $x_1 = 1, x_2 = \frac{-1 \pm \sqrt{-3}}{2}, x_3 = -x_2 - 1$; $\mathcal{A}_2 = 0$ 有两组解: $x_1 = \frac{-1 \pm \sqrt{-3}}{2}, x_2 = -1, x_3 = -x_1 - 1$;

$\mathcal{A}_3 = 0$ 有两组解：$x_1 = \frac{-1\pm\sqrt{-3}}{2}, x_2 = -x_1 - 1, x_3 = 1$. 因而, $F_3 = 0$ 有六组解.

3. 吴升列形式的零点分解. 为了提高计算效率, 我们使用了其他形式的升列和相关的计算方法. 本节介绍由吴文俊给出的一种改进形式. 对于非零多项式 P, Q, 我们称 P 对 Q 是吴约化的, 如果当 $\mathrm{cls}(P) \neq \mathrm{cls}(Q)$ 时, $\mathrm{I}(P)$ 对 Q 是约化的; 当 $\mathrm{cls}(P) = \mathrm{cls}(Q)$ 时, P 对 Q 是约化的.

一个多项式三角列 $\mathcal{A} : A_1, A_2, \cdots, A_r$ 称作是一个吴升列, 如果对任意的 $i < j$, A_j 对 A_i 是吴约化的. 多项式 P 称为对吴升列 $\mathcal{A}$ 是吴约化的, 如果 P 对 $\mathcal{A}$ 中的每个多项式都是吴约化的.

算法 wuprem 给出多项式 F 对多项式 G 的吴余式算法：即给定多项式 F, G 和变元 x, 存在非负整数 s 满足 $\mathrm{lc}(G, x)^s F = QG + R$, 其中 Q, R 为 $\mathbb{K}[\mathbb{X}]$ 上的多项式, 并且 R 还满足：如果 $\mathrm{cls}(R) = x$, 则 $\deg(R, x) < \deg(G, x)$. 如果 $\mathrm{cls}(R) \neq x$, 则 $\deg(\mathrm{I}(R), x) < \deg(G, x)$. 由此知道, 如果 x 是 G 的主变元, 那么 R 对 G 是吴约化的.

Algorithm 7: wuprem(F, G, x)

Input: 多项式 F, G, 变元 x

Output: R: 为 F 对 G 的吴余式

$R = F$

if $\mathrm{cls}(F) = x$ **then return** $\mathrm{prem}(F, G, x)$

I=I(F)

R=red(F)

if $\deg(I, x) < \deg(G, x)$ **then return** R

$lc_g = \mathrm{lc}(G, x)$

while $\deg(I, x) \geqslant \deg(G, x)$ **do**

  $lc_r = \mathrm{lc}(I, x)$

  $deg_r = \deg(I, x)$

  $I = lc_g I - lc_r G x^{(deg_r - deg_g)}$

  $R = lc_g R - lc_r G x^{(deg_r - deg_g)}$

end

if $I = 0$ **then**

  return wuprem(R, G, x)

else

  $y = \mathrm{lvar}(F)$

 return $I y^{\deg(F, y)} + R$

end

只需将约化与余式的概念变为吴约化与吴余式, 前面给出的零点分解定理就可以用来计算关于吴升列的零点分解.

定理 4.3.3 *存在一个算法, 对于给定的多项式集 $\mathbb{P}$, 可在有限步内得到有限个吴升列 $\mathcal{C}_i$, 使得*

$$\text{Zero}(\mathbb{P}) = \cup_i \text{Zero}(\mathcal{C}_i/I_{\mathcal{C}_i})$$

函数 wsolve 中的参数 type 为 “wu” 时, 输出的升列是吴升列.

```
>wsolve([x3+x2+x1,x3*x2+x2*x1+x1*x3,x3*x2*x1-1],[x3,x2,x1],[],"wu");
[[x3+x2+x1, x2^2+x1*x2+x1^2, x1-1],
 [x3+x2+x1, x2^2+x1*x2+x1^2, x1^2+x1+1]]
>wsolve([x4+x3+x2+x1, x4*x3+x3*x2+x2*x1+x1*x4,
        x4*x3*x2+x3*x2*x1+x2*x1*x4+x1*x3*x4, x4*x3*x2*x1-1],
        [x4, x3, x2, x1], [], "wu");
[[x4+x3+x2+x1, x3+x1, x1*x2+1],
 [x4+x3+x2+x1, x3+x1, x1*x2-1],
 [x4+x2, x3+x1, x1*x2+1],
 [x4+x2, x3+x1, x1*x2-1]]
```

与上一节的结果相比, 这里给出的结果在形式上是不同的. 对于第一个例子, 我们得到两个升列. 这是因为第二个升列中没有对 $x_1^2 + x_1 + 1$ 做除法, 因此多项式 $x_2^2 + x_1 * x_2 + x_1^2$ 是不可约的. 使用吴升列的好处是减少计算量, 但是我们也看到这样给出的升列不总是最简形式. 根据经验, 对于规模较大的问题使用吴升列可以提高计算速度. 下面几种升列也有这样的特点. 对于第二个例子升列 $[x_4 + x_3 + x_2 + x_1, x_3 + x_1, x_1 * x_2 + 1]$ 与 $[x_4 + x_2, x_3 + x_1, x_1 * x_2 + 1]$ 实际上是相同的. 由于没有简化, 两个升列都被给出.

4. 弱升列形式的零点分解. 一个多项式三角列 $\mathcal{A} : A_1, A_2, \cdots, A_r$ 称作是一个*弱升列*, 如果对 $i > 2$, $\text{I}(A_i)$ 关于 $\mathcal{A}_{i-1} : A_1, \cdots, A_{i-1}$ 的余式不为零. 多项式 P 称为对弱升列 $\mathcal{A}$ 是*弱约化*的如果满足下面两个条件：(1) P 的初式对 $\mathcal{A}$ 的余式不为 0; 即 $\text{premas}(\text{I}(P), \mathcal{A}) \neq 0$; (2) 如果存在 i, 有 $\text{cls}(A_i) = \text{cls}(P)$, 则 P 对 A_i 是约化的.

在这种弱升列形式下, 我们将给出多项式 P 对弱升列 $\mathcal{A}$ 的余式算法. 见算法 wpremas.

对于弱升列, 同样有零点分解定理. 在 MMP 实现的所有零点分解定理中, 下面的弱升列形式是对升列形式要求最弱的一种. 其他形式的升列都是弱升列.

定理 4.3.4 *存在一个算法, 对于给定的多项式集 $\mathbb{P}$, 可在有限步内得到有限个弱升列 $\mathcal{C}_i$, 使得*

$$\text{Zero}(\mathbb{P}) = \cup_i \text{Zero}(\mathcal{C}_i/I_{\mathcal{C}_i})$$

Algorithm 8: wpremas($P, \mathcal{A}$)

```
Input: 多项式 P, 弱升列 A : A_1, A_2, ..., A_r.
Output: R, 为 P 对弱升列 A 的余式, R 对 A 是弱约化的.
R = P
i = r.
while i ≠ 0 do
    x =lvar(A_i);
    y =lvar(R)
    if y > x & premas(I(R),A)=0 then return premas(R, A)
    if y = x then R =prem(R, A_i, x)
    i = i − 1.
end
return R
```

在 MMP 系统中, 函数 wsolve 中的参数 type 时 “weak” 时, 输出的升列是弱升列. 下例中的 F 在第 80 页给出.

```
>wsolve(F, [z, y, x], [], "weak");
[[x^2-2*z*x+5, y, x^2+5],
 [x^2-2*z*x+5, x^6+8*y*x^4+15*x^4+75*x^2+125, 3*x^6-64*x^5
 +45*x^4+225*x^2+375],
 [x^2-2*z*x+5, x^6+8*y*x^4+15*x^4+75*x^2+125, x^2+5]]
```

5. 正则升列形式的零点分解. 设 $\mathcal{A}$ 是一个形如 (4.3) 的升列. 一个多项式 P 对一个升列 $\mathcal{A}$ 是可逆的, 如果 $f \in \mathbb{K}[\mathbb{U}]$ 或者 $(A_1, \cdots, A_k, f) \cap \mathbb{K}[\mathbb{U}] \neq \{0\}$, 其中 k 满足 $\mathrm{lvar}(P) = y_k$.

升列 $\mathcal{A} : A_1, \cdots, A_p$ 称为是 *正则升列*, 如果 $\mathrm{I}(A_i), i = 2, \cdots, p$ 关于 $\mathcal{A}$ 可逆. 正则升列有下面的性质 ([4]):

引理 4.3.5　*$\mathcal{A}$ 是一个正则升列当且仅当 $\mathcal{A}$ 是其饱和理想 $\mathrm{sat}(\mathcal{A})$ 的基列.*

设 $\mathcal{P}$ 是一个多项式集合, 我们用 $(\mathcal{P})$ 表示 $\mathcal{P}$ 中的多项式在 $\mathbb{K}[\mathbb{X}]$ 中生成的理想. 对于正则升列, 由上述引理, 我们知道：设 $\mathcal{A}$ 是一个正则升列, 所有对 $\mathcal{A}$ 求余余式为 0 的多项式是一个理想. 即

$$I = \{P | P \in \mathbb{K}[\mathbb{X}], \mathrm{premas}(P, \mathcal{A}) = 0\}$$

I 是一个理想, 而且 $I = \mathrm{sat}(\mathcal{A})$.

引理 4.3.6　*多项式 P 对正则升列 $\mathcal{A}$ 是可逆的当且仅当 $\mathrm{Res}(P, \mathcal{A}) \neq 0$.*

引理 4.3.7　*设 $\mathcal{A}$ 是正则升列, F 是一个对 $\mathcal{A}$ 约化的非零多项式. 如果 $\mathrm{Res}(F, \mathcal{A}) = 0$. 则存在对 $\mathcal{A}$ 已约化的非零多项式 G, 使得*

$$\text{Zero}(\mathcal{A}) = \text{Zero}(\mathcal{A} \cup \{F\}) \cup \text{Zero}(\mathcal{A} \cup \{G\})$$

文献 [6] 给出了利用 Gröbner 基计算上述 G 的一个算法，在 [76] 中，一种不需要计算 Gröbner 基的方法也可以给出类似的结果. 见算法 invert.

Algorithm 9: $invert(F, \mathcal{A})$

Input: 多项式 F 和正则升列 $\mathcal{A}$

Output: $(test, G)$ 满足: $test = true$ 如果 F 对 $\mathcal{A}$ 是可逆的, 并且 $G = 0$; $test = false$ 如果 F 对 $\mathcal{A}$ 是不可逆的, 对 $\mathcal{A}$ 约化的非零多项式 G, 而且 $FG \in (\mathcal{A})$

W= 一个新变元

U=$\mathcal{A}$ 的参变元

Y=$\mathcal{A}$ 中多项式出现的所有主变元

$P = (\mathcal{A}, F) \cap \mathbb{K}[\mathbb{U}, \mathbb{W}]$ 在序 $U < W < Y\pi$ 的 Gröbner 基

if $P(U, 0) \neq 0$ **then**

  test=true; G=0

else

  test=false

  $G = \text{premas}\left(\dfrac{P(U,F)}{F}, \mathcal{A}\right)$

end

return $(test, G)$

算法 regdec 可以得到如下形式的零点分解定理.

定理 4.3.8 设 $\mathbb{P}$ 为一个多项式组, 在有限步内可以得到有限个正则升列 $\mathcal{A}_i$, 使得有下列形式的零点分解

$$\text{Zero}(\mathbb{P}) = \cup_i \text{Zero}(\mathcal{A}_i / I_{\mathcal{A}_i})$$

如果系统 $\mathbb{P}$ 只有有限个零点, 则我们有

$$\text{Zero}(\mathbb{P}) = \cup_i \text{Zero}(\mathcal{A}_i)$$

在 MMP 中通过设置 wsolve 的第四个参数 type 为 “regular” 来得到具有正规升列形式的零点分解.

```
>G:=[(1-x3)^2+(1-x3)^2-x1^2,(1-x3)^2+(x2-x3)^2-x1^2,
    (x3-x4)^2+(x5-x3)^2-x1^2, (1-x4)^2+(x2-x5)^2-x1^2,
    (x4-x1)^2+x5^2-x1^2,   2*(x4-x5)^2-x1^2]:
>ord:=[x5, x4, x3, x2, x1]:
>wsolve(G, ord,[] "regular");
[[x1^7+2*x5*x1^6-4*x1^6+8*x5*x1^5-50*x1^5+16*x5*x1^4+120*x1^4
```

```
-92*x5*x1^3-80*x1^3+104*x5*x1^2-4*x1^2-56*x5*x1+24*x1+16*x5-8,
3*x4*x1^7+2*x1^7+4*x4*x1^6-15*x1^6-34*x4*x1^5+33*x1^5+28*x4*x1^4
-17*x1^4+24*x4*x1^3+2*x1^3-60*x4*x1^2-16*x1^2+40*x4*x1+16*x1
-8*x4-4, 2*x1^4*x3+12*x1^3*x3-42*x1^2*x3+44*x1*x3-20*x3-x1^5
-12*x1^4+18*x1^3+12*x1^2-32*x1+16, x1^5-x2*x1^4+11*x1^4
-6*x2*x1^3-24*x1^3+21*x2*x1^2+9*x1^2-22*x2*x1+10*x1+10*x2-6,
x1^8+8*x1^7-22*x1^6+20*x1^5+18*x1^4-24*x1^3-24*x1^2+32*x1-8]]
```

Algorithm 10: regdec($\mathbb{P}$)

Input: 多项式集合 $\mathbb{P}$

Output: 有限个正则升列 $\mathcal{A}_i$, 使得 $\text{Zero}(\mathbb{P}) = \cup_i \text{Zero}(\mathcal{A}_i / I_{\mathcal{A}_i})$

$\mathcal{C}$ =charset($\mathbb{P}$)=$C_1, \cdots, C_r$

if $r = 1$ **then**

　RETURN $\mathcal{C} \bigcup \cup_{i=1}^{r} \text{regdec}(\mathbb{P} \cup \mathcal{C} \cup \{I(C_1)\})$

else

　$\mathcal{A} = C_1$

　$i = 2$

　while $i \leqslant r$ **do**

　　F =I(A_i)

　　$(test, G) = invert(F, \mathcal{A})$

　　if $test$ **then**

　　　$\mathcal{A} = \mathcal{A} \cup \{C_i\}$

　　else

　　　return regdec($\mathbb{P} \cup \mathcal{C} \cup \{F\}$)$\cup$regdec($\mathbb{P} \cup \mathcal{C} \cup \{G\}$)

　　end

　　$i = i + 1$

　end

　return $\{\mathcal{C}\} \bigcup \cup_{i=1}^{r} \text{regdec}(\mathbb{P} \cup \mathcal{C} \cup \{I(C_i)\})$

end

6. 正规升列形式的零点分解.　升列 $\mathcal{A}$ 称为是正规升列, 如果 $\mathcal{A}$ 中多项式的初式不含 $\mathcal{A}$ 中多项式的主变元.

设 P, Q 是两个关于 x 的多项式, $R = \text{Res}(P, Q, x)$. 则可以计算出多项式 F, G 满足

$$FP + GQ = R$$

其中 $\deg(F, x) < \deg(Q, x)$, $\deg(G, x) < \deg(P, x)$. 算法 $\text{Res}(P, Q, x)$ 将给出 (F, G, R), 在这儿我们就不具体给出这个算法. 设 $\mathcal{A} : A_1, \cdots, A_r$ 是一个升列, A_i 的主变元为

y_i. 则定义: $\mathrm{Res}(P,\mathcal{A})=\mathrm{Res}(\mathrm{Res}(P,A_r,y_r),A_1,\cdots,A_{r-1})$.

为了将正则升列正规化, 我们首先给出一个简单情形的算法: $\mathcal{A}:A_1,\cdots,A_r$ 是一个正则升列, 设 $A_1,\cdots,A_r$ 的主变元分别为 $y_1,\cdots,y_r$, 而 $\mathcal{A}_{r-1}:A_1,\cdots,A_{r-1}$ 是一个正规升列. 算法 normalizationl 给出一个正规化算法.

Algorithm 11: normalizationl($\mathcal{A}$)

Input: 正则升列 $\mathcal{A}:A_1,\cdots,A_r$
Output: 正规升列 $\mathcal{A}'$: 正则升列 $\mathcal{A}$ 的正规化升列.
if $r=1$ **then return** $\mathcal{A}$ $d=\deg(A_r,y_r)$
R_{r-1}=I(A_r), $i=r-1,C:=1$
while i >0 **do**
  (F_i,G_i,R_{i-1})=Res(R_i,A_i,y_i)
  $C:=CF_i$
end
$I:=R_0$
$A_r':=Iy_r^d+C\mathrm{red}(A_r)$
A_r'' =premas$(A_r',\mathcal{A}_{r-1})$
return $\mathcal{A}_{r-1}\cup\{A_r'\}$

有了算法 normalizationl, 我们很容易将一个正则升列转换成一个正规升列. 设 $\mathcal{A}:A_1,\cdots,A_r$ 是一个正则升列, 算法 normalization 将它正规化.

Algorithm 12: normalization($\mathcal{A}$)

Input: 正则升列 $\mathcal{A}:A_1,\cdots,A_r$
Output: 正规升列 $\mathcal{A}'$: 正则升列 $\mathcal{A}$ 的正规化升列.
if $r=1$ **then**
  return $\mathcal{A}$
else
  $\mathcal{A}'_1:=A_1$
  $i:=2$
  while $i\leqslant r$ **do**
    $\mathcal{A}_i:=\mathcal{A}'_{i-1}\cup\{A_i\}$
    $\mathcal{A}'_i$:=normalization1$(\mathcal{A}_i)$
  end
  return $\mathcal{A}'_r$
end

定理 4.3.9 设 $\mathbb{P}$ 为一个有限多项式组, 在有限步内可以得到有限个正规升列 $\mathcal{A}_i$,

使得有下列形式的零点分解成立

$$\text{Zero}(\mathbb{P}) = \cup_i \text{Zero}(\mathcal{A}_i / I_{\mathcal{A}_i})$$

如果系统 $\mathbb{P}$ 只有有限个零点, 则我们有

$$\text{Zero}(\mathbb{P}) = \cup_i \text{Zero}(\mathcal{A}_i)$$

在 MMP 中通过设置 wsolve 的第四个参数 type 为 “normal” 来得到具有正规升列形式的零点分解.

```
>H:=[x4+x3+x2+x1,x3*x4+x2*x3+x1*x2,x2*x3*x4+
     x1*x3*x4+x1*x2*x4+x1*x2*x3,x1*x2*x3*x4-1];
>ord:=[x4, x3, x2, x1];
>wsolve(H, ord, ord, "normal");
[[x1+x4, 2*x1+x3, 2*x1-x2, 2*x1^2+1],
 [x1+x4, 2*x1+x3, 2*x1-x2, 2*x1^2-1],
 [-x1^3+x1-x4, x1^3-x1-x3, x1+x2, x1^4-x1^2+1],
 [x1-x4-1, -x1-x3+1, x1+x2, x1^2-x1+1],
 [-x1+x4-1, x1+x3+1, x1+x2, x1^2+x1+1]]
```

7. 饱和升列形式的零点分解. 一个正则升列 $\mathcal{A}$ 称为是饱和升列, 如果它的隔离子关于 $\mathcal{A}$ 是可逆的. 饱和升列也称为无平方升列. 由算法 satdec 可以计算多项式饱和形式的零点分解.

定理 4.3.10 设 $\mathbb{P}$ 为一个多项式组, 在有限步内可以得到有限个饱和升列 $\mathcal{A}_i$, 使得有下列形式的零点分解

$$\text{Zero}(\mathbb{P}) = \cup_i \text{Zero}(\mathcal{A}_i / I_{\mathcal{A}_i})$$

如果系统 $\mathbb{P}$ 只有有限个零点, 则我们有

$$\text{Zero}(\mathbb{P}) = \cup_i \text{Zero}(\mathcal{A}_i)$$

实际上我们需要结合算法 satdec和算法 scs来提高计算的效率, 而且同样可以得到具有上述形式的零点分解.

在 MMP 中通过设置 wsolve 的第四个参数 type 为 “saturated” 来得到具有饱和形式的零点分解.

```
>wsolve([x^2+1, y^2-2*x*y-1], [y, x], [], "saturated");
[[x-y, x^2+1]]
```

注意到 $\text{S}(y^2 - 2*x*y - 1) = 2(y - x)$, 而 $y^2 - 2*x*y - 1 = (y - x)^2 - (x^2 + 1)$. 所以, 上例中的输入升列不是饱和的. wsolve 给出了相应的饱和升列.

8. 不可约升列形式的零点分解. 设 $\mathcal{A}$ 是一个形如 (4.3) 的升列. 设 $\mathbb{K}_0=\mathbb{K}(u_1,\cdots,u_q)$, 即 $\mathbb{K}_0$ 是系数在 $\mathbb{K}$ 上的关于 $u_1,\cdots,u_q$ 的有理函数域. 如果 A_1 作为 $\mathbb{K}_0[y_1]$ 上的多项式不可约, 不妨设 η_1 是不可约多项式 A_1 的一个零点. 设 $\mathbb{K}_1=\mathbb{K}_0(\eta_1)$, 如果 $A_2'=A_2|_{y_1=\eta_1}$ 作为 $\mathbb{K}_1[y_2]$ 上的多项式在 $\mathbb{K}_1$ 上不可约, 设 η_2 是 A_2' 的一个零点. 如此下去, $A_i'=A_i|_{y_1=\eta_1,\cdots,y_{i-1}=\eta_{i-1}}$ 作为 $\mathbb{K}_{i-1}[y_i]$ 上的多项式在 $\mathbb{K}_{i-1}$ 上不可约. 而 η_i 是 A_i 的一个零点, $i=2,\cdots,p$. 这样的升列 $\mathcal{A}$ 称为一个*不可约升列*. 而 $(u_1,\cdots,u_q,\eta_1,\cdots,\eta_p)$ 称为不可约升列 $\mathcal{A}$ 的一个*母点*. 有时为方便起见, 我们将 $(u_1,\cdots,u_q,\eta_1,\cdots,\eta_p)$ 按照 $x_1,\cdots,x_n$ 的顺序重新写成 $(\xi_1,\cdots,\xi_n)$. 我们也称 $\xi=(\xi_1,\cdots,\xi_n)$ 是不可约升列 $\mathcal{A}$ 的一个母点.

Algorithm 13: satdec($\mathbb{P}$)

Input: 多项式集合 $\mathbb{P}$
Output: 有限个饱和升列组成的集合 $\mathcal{A}_i$, 使得
  Zero($\mathbb{P}$)=$\cup_i$Zero($\mathcal{A}_i/I_{\mathcal{A}_i}$)
$\mathcal{C}$=charset($\mathbb{P}$)(假设 $\mathcal{C}=C_1\cdots,C_r$)
if $r=1$ **then**
  return $\mathcal{C}\cup$satdec($\mathbb{P}\cup\mathcal{C}\cup\{\mathrm{I}(C_1)\}$)$\cup$satdec($\mathbb{P}\cup\mathcal{C}\cup\{\mathrm{S}(C_1)\}$)
else
  $\mathcal{A}=\{C_1\},\ i=2$
  while $i\leqslant r$ **do**
    if $\mathrm{I}(C_i)$ *对* $\mathcal{A}$ *是可逆的* **then**
      $\mathcal{A}=\mathcal{A}\cup\{C_i\}$
      if $\mathrm{S}(C_i)$ *对* $\mathcal{A}$ *不是可逆的* **then**
        G' =FactorAS(S(C_i),$\mathcal{A}$)
        return satdec($\mathbb{P}\cup\mathcal{C}\cup\{\mathrm{S}(C_i)\}$)$\cup$satdec($\mathbb{P}\cup\mathcal{C}\cup\{G'\}$)
      end
    else
      G =FactorAS(I(A_i), $\mathcal{A}$)
      return satdec($\mathbb{P}\cup\mathcal{C}\cup\{\mathrm{I}(A_i)\}$)$\cup$ satdec($\mathbb{P}\cup\mathcal{C}\cup\{G\}$)
    end
    $i=i+1$
  end
  return $\{\mathcal{C}\}\bigcup\cup_{i=1}^{r}$satdec($\mathbb{P}\cup\mathcal{C}\cup\{I(C_i)\}$)
end

算法 algfactor 给出多项式 $F\in\mathbb{Q}[x_1,\cdots,x_n,y]$ 在不可约升列 $\mathcal{A}:A_1,\cdots,A_r$, $\mathrm{cls}(A_i)=p_i$ 定义的扩域上的因式分解.

Algorithm 14: algfactor$(F,\mathcal{A})$

Input: 无平方因子多项式 $F\in\mathbb{Q}[x_1,\cdots,x_n,y]$, 不可约升列
$\mathcal{A}:A_1,\cdots,A_r\subset\mathbb{Q}[x_1,\cdots,x_n]$. 设 $p_i=\mathrm{cls}(A_i)$.

Output: F 的因式分解

if $\deg(F,y)\leqslant 1$ **then return** F

repeat

随机选一组整数 $[a_1,\cdots,a_{r-1}]$

Q_1=Res$(y-x_{pr}-a_{r-1}x_{p_{r-1}}-\cdots-a_1x_{p1},\mathcal{A})$

until Q_1 是 $\mathbb{Q}$ 上关于 y 的不可约多项式;

repeat

选随机整数 $c; c_i=c*a_i(1\leqslant i\leqslant r-1), c_r=c$.

$f'=f|_{y=y+c_rx_{pr}+\cdots+c_1x_{p1}}$

$R(y)=\mathrm{Res}(f',\mathcal{A})$

until R 关于 y 无平方因子;

$G'=R$ 在 $\mathbb{Q}$ 上所有包含变元 y 的因式.

$G=\emptyset$.

for $g\in G'$ do

$f'_g=GCD(f',g)$(在升列 $\mathcal{A}$ 定义的扩域上)

$f_g=f'_g|_{y=y-c_rx_{pr}-\cdots-c_1x_{p1}}$

if $\deg(f_g,y)\geqslant 1$ **then** $G=G\cup\{f_g\}$

return G.

引理 4.3.11 如果 $\mathcal{A}$ 是一个不可约升列, 则 $\mathrm{sat}(\mathcal{A})$ 是一个素理想. $\mathcal{A}$ 的母点也是 $\mathrm{sat}(\mathcal{A})$ 的母点, 等价地, 对于多项式 P, 我们有

$$\mathrm{premas}(P,\mathcal{A})=0\Leftrightarrow P(\xi)=0$$

对于由不可约升列定义的扩域上的单变元的多项式的最大公因式, 可以用经典的欧基里德算法, 即辗转相除法求得. 下面我们介绍多项式环 $\mathbb{Q}(\xi)[y]$ 中的多项式的因式分解算法, 其中 $\mathbb{Q}$ 是有理数域, ξ 由不可约升列 $\mathcal{A}$ 所定义. 给定多项式 $F\in\mathbb{Q}[x_1,\cdots,x_n,y]$, F 在不可约升列 $\mathcal{A}:A_1,\cdots,A_r$ 所定义的扩域中可分解为 $F_1,\cdots,F_t$, 即存在非负整数 s 和多项式 $D\in K[x_1,\cdots,x_n]$, $\mathrm{premas}(D,\mathcal{A})\neq 0$, $P_i\in\mathbb{Q}[x_1,\cdots,x_n,y], i=1,\cdots,t$ 使得

$$I_{\mathcal{A}}^s(DF-F_1\cdots F_t)=\sum_{i=1}^r P_iA_i$$

其中 $\deg(F_i,y)\neq 0$ 对 $i=1,\cdots,t$.

更进一步，可以使得 F 在由不可约升列 $\mathcal{A}$ 所定义的扩域中的因式 F_i 对 $\mathcal{A}$ 是约化的. 在计算扩域上多项式的因式分解的基础上, 我们可以得到多项式系统的不可约形式的零点分解.

给定一个升列 $\mathcal{A}: A_1, \cdots, A_{r-1}, A_r$, 设 $\mathcal{A}': A_1, \cdots, A_{r-1}$ 是不可约升列, 那么算法 irrfactor1 计算一组不可约升列 $\mathcal{A}_i$, $i = 1, \cdots, s$, 使得

$$\begin{gathered}\text{Zero}(\text{sat}(\mathcal{A})) = \cup_i \text{Zero}(\text{sat}(\mathcal{A}_i)) \\ \cup_i \text{Zero}(\mathcal{A}_i / I_{\mathcal{A}_i}) \subset \text{Zero}(\mathcal{A}/I_{\mathcal{A}}) \subset \cup_i \text{Zero}(\mathcal{A}_i)\end{gathered} \tag{4.6}$$

Algorithm 15: irrfactor1($\mathcal{A}$)

Input: 升列 $\mathcal{A}: A_1, \cdots, A_r$, 而且 $\mathcal{A}': A_1, \cdots, A_{r-1}$ 是不可约升列

Output: 一组不可约升列 $\mathcal{A}_i$, 满足 4.6

$I = I(A_r)$

if premas$(I, \mathcal{A}') = 0$ **then**

  return $\emptyset$

else

  $R = \{F|F$ 是 A_r 在 $\mathcal{A}'$ 定义的域中的因式且 cla(F)=cls$(A_r)\}$

  return $\cup_{F \in R}(\mathcal{A}', F)(\mathcal{A}'' = \mathcal{A}', F$ 是一个不可约升列)

end

对于升列 $\mathcal{A}$, 算法 irrfactor 计算不可约升列 $\mathcal{A}_i$, 使得

$$\begin{gathered}\cup_i \text{Zero}(\mathcal{A}_i / I_{\mathcal{A}_i}) \subset \text{Zero}(\mathcal{A}/I_{\mathcal{A}}) \subset \cup_i \text{Zero}(\mathcal{A}_i) \\ \text{Zero}(\mathcal{A}/I_{\mathcal{A}}) = \cup_i \text{Zero}(\text{sat}(\mathcal{A}_i)/I_{\mathcal{A}})\end{gathered} \tag{4.7}$$

Algorithm 16: irrfactor($\mathcal{A}$)

Input: 升列 $\mathcal{A}: A_1, \cdots, A_r$

Output: 一组不可约升列 $\mathcal{A}_i$, 满足 4.7

if $r = 1$ **then**

  $R = \{F|F$ 是 A_1 的因子且 cls(F) =cls$(A_1)\}$

  return $\cup_{F \in R}\{F\}$

else

  $A' = A_1, \cdots, A_{r-1}$

  B =irrfactor$(\mathcal{A}')$

  return $\cup_{\mathcal{B} \in B}$irrfactor1$(\{\mathcal{B}, \mathcal{A}_r\})$

end

而且 $I_{\mathcal{A}}$ 对所有的 $\mathcal{A}_i$ 的余式非零.

实际上我们需要结合算法 irrfactor和算法 scs来提高计算的效率, 而且同样可以得到下面形式的零点分解. 算法 irrdec 计算代数簇的不可约分解.

Algorithm 17: irrdec($\mathbb{P}$)

Input: 一个多项式集合
Output: 一组不可约升列 $\mathcal{A}_i$, 满足 $\mathrm{Zero}(\mathbb{P}) = \cup_i \mathrm{Zero}(\mathcal{A}_i / I_{\mathcal{A}_i})$
$A = \mathrm{scs}(\mathbb{P})$
$B = \emptyset$
while $\mathcal{A} \in A$ **do**
  $B = B \cup$ irrfactor($\mathcal{A}$)
end
return $B \bigcup \cup_{\mathcal{A} \in B} \mathrm{irrfactor}(\mathbb{P} \cup \mathcal{A} \cup \{I_{\mathcal{A}}\})$

定理 4.3.12 *设 $\mathbb{P}$ 为一个有限多项式集合. 则在可以得到有限个不可约升列 $\mathcal{A}_i$, 使得有下列形式的零点分解*

$$\mathrm{Zero}(\mathbb{P}) = \cup_i \mathrm{Zero}(\mathcal{A}_i / I_{\mathcal{A}_i})$$

如果系统 $\mathbb{P}$ 只有有限个零点, 则我们有

$$\mathrm{Zero}(\mathbb{P}) = \cup_i \mathrm{Zero}(\mathcal{A}_i)$$

9. 代数簇的不可缩分解. 本节将给出定理 4.2.1 的构造性证明, 即对于有限多项式集合 $\mathbb{P}$, 将 $\mathrm{Zero}(\mathbb{P})$ 分解为互不包含的不可约代数簇的并.

将定理 4.2.7中的最后一个等式做适当的修改得到

$$\mathrm{Zero}(\mathbb{P}) = \mathrm{Zero}(\mathrm{sat}(\mathcal{C})) \bigcup \cup_{i=1}^{r} \mathrm{Zero}(\mathbb{P} \cup \mathcal{C} \cup \{\mathrm{I}(C_i)\})$$

需要注意到 $\mathrm{I}(C_i)$ 对 $\mathcal{C}$ 已约化, 因此, $\mathbb{P} \cup \mathcal{C} \cup \mathrm{I}(C_i)$ 的基列的秩低于 $\mathcal{C}$ 的秩. 对每一个 $\mathbb{P} \cup \mathcal{C} \cup \mathrm{I}(C_i)$, 再递归调用整序原理的过程. 由引理 4.2.4, 这样的递归过程在有限步内一定终止. 由上面的讨论, 我们可以假定所得到的升列是不可约的. 因此我们得到下面的定理.

定理 4.3.13 (代数簇分解算法) *存在一个算法, 对于给定的多项式集 $\mathbb{P}$, 或者判定 $\mathrm{Zero}(\mathbb{P}) = \emptyset$, 或者可在有限步内得到有限个不可约升列 $\mathcal{C}_i$, 使得*

$$\mathrm{Zero}(\mathbb{P}) = \cup_i \mathrm{Zero}(\mathrm{sat}(\mathcal{C}_i)) \tag{4.8}$$

算法 irrvardec 给出另一个计算代数簇的不可约分解的算法.

Algorithm 18: irrvardec($\mathbb{P}$)

Input: 一个多项式集合

Output: 一组不可约升列 $\mathcal{A}_i$, 满足 $\mathrm{Zero}(\mathbb{P}) = \cup_i \mathrm{Zero}(\mathrm{sat}(\mathcal{A}_i))$

由零点分解得到一组升列 $\mathcal{A}_i$, $i = 1, \cdots, r$ 满足

$\mathrm{Zero}(\mathbb{P}) = \cup_i \mathrm{Zero}(\mathcal{A}_i / I_{\mathcal{A}_i})$

对每一个升列 $\mathcal{A}_i$ 用 irrfactor 进行分解, 得到一组升列 $\mathcal{A}_{ij}, j = 1, \cdots, n_i$

满足 $\mathrm{Zero}(\mathcal{A}_i) = \cup_{j=1}^{n_i} \mathrm{Zero}(\mathrm{sat}(\mathcal{A}_{ij}) / I_{\mathcal{A}_i})$

return $\cup_{i=1}^{r} \cup_{j=1}^{n_i} \mathcal{A}_{ij}$

下面我们证明该算法的正确性. 首先我们给出一个引理.

引理 4.3.14 *设 $\mathcal{A}$ 是一个不可约升列, P, Q 是多项式, 满足 $\mathrm{Zero}(\mathrm{sat}(\mathcal{A})/Q) \subset \mathrm{Zero}(P)$, 而且 $\mathrm{premas}(Q, \mathcal{A}) \neq 0$, 则 $\mathrm{Zero}(\mathrm{sat}(\mathcal{A})) \subset \mathrm{Zero}(P)$.*

证明 设 ξ 是不可约升列 $\mathcal{A}$ 的母点, 那么 $I_{\mathcal{A}}(\xi) \neq 0$, 又因为 $\mathrm{premas}(Q, \mathcal{A}) \neq 0$, 因此 $Q(\xi) \neq 0$. 那么 $\xi \in \mathrm{Zero}(\mathrm{sat}(\mathcal{A})/Q) \subset \mathrm{Zero}(P)$, 即 $P(\xi) = 0$, 因此 $\mathrm{Zero}(\mathrm{sat}(\mathcal{A})) \subset \mathrm{Zero}(P)$.

对多项式组 $\mathbb{P}$, 由零点分解算法, 得到升列 $\mathcal{A}_i$, 使得 $\mathrm{Zero}(\mathbb{P}) = \cup_i \mathrm{Zero}(\mathcal{A}_i / I_{\mathcal{A}_i})$, 而对每一个 $\mathcal{A}_i$, 则由算法 irrfactor, 可以得到它的一组不可约升列形式 $\mathcal{A}_{ij}$, 满足

$$\mathrm{Zero}(\mathcal{A}_i / I_{\mathcal{A}_i}) = \cup_j \mathrm{Zero}(\mathrm{sat}(\mathcal{A}_{ij}) / I_{\mathcal{A}_i})$$

而且 $I_{\mathcal{A}_i}$ 对任意的 $\mathcal{A}_{ij}$ 的余式非零. 因此有

$$\mathrm{Zero}(\mathbb{P}) = \bigcup_i \cup_j \mathrm{Zero}(\mathrm{sat}(\mathcal{A}_{ij}) / I_{\mathcal{A}_i})$$

由于每一个 $\mathcal{A}_{ij}$ 是不可约升列, 根据上面的引理, 知道

$$\mathrm{Zero}(\mathbb{P}) = \bigcup_i \cup_j \mathrm{Zero}(\mathrm{sat}(\mathcal{A}_{ij}))$$

在 MMP 中通过设置 wsolve 的第四个参数 type 为 “irr” 或 “irreducible” 来得到具有不可约形式的零点分解.

给定不可约升列 $\mathcal{A}$ 的一个母点 ξ, 下面的多项式集合

$$\{P | P \in \mathbb{K}[x_1, \cdots, x_n], P(\xi) = 0\}$$

确定了 $\mathbb{K}[x_1, \cdots, x_n]$ 中的一个素理想. 这个素理想称为不可约升列 $\mathcal{A}$ 或母点 ξ 所确定的理想. 实际上, 这个素理想就是升列 $\mathcal{A}$ 的饱和理想 $\mathrm{sat}(\mathcal{A})$. 因此, 我们知道 $\mathbb{K}[\mathbb{X}]/\mathrm{sat}(\mathcal{A})$ 是一个可除环, 其分式域与 $\mathbb{K}(\xi)$ 同构. 下面给出了计算上面的素理想的基的一个方法.

设 $\mathcal{A}$ 是一个不可约升列, 令 $\mathcal{P} = \mathcal{A} \cup I_{\mathcal{A}}z - 1$, z 是一个新的未定元. 设 G 是 $\mathbb{P}$ 在变元序 $z > x$ 下的 Gröbner 基, 那么 $G \cap \mathbb{K}[\mathbb{X}]$ 就是 $\mathcal{A}$ 所确定的理想的一组基.

考虑公式 (4.8) 中的 $\mathcal{C}_i$. 由上面的算法可以计算出素理想 $\mathrm{sat}(\mathcal{C}_i)$ 的 Gröbner 基 B_i. 显然有

引理 4.3.15 $\mathrm{Zero}(\mathrm{sat}(\mathcal{C}_i)) \subset \mathrm{Zero}(\mathrm{sat}(\mathcal{C}_j))$ 当且仅当对 $P \in \mathcal{C}_j$, $\mathrm{premas}(P, \mathcal{C}_i) = 0$.

由上述引理可以去掉公式 (4.8) 中多余的分支, 设剩余的分支为 $\mathcal{A}_1, \cdots, \mathcal{A}_s$. 最后得到下列不可缩分解

$$\mathrm{Zero}(\mathbb{P}) = \cup_i \mathrm{Zero}(\mathrm{sat}(\mathcal{A}_i)) = \cup_i \mathrm{Zero}(\mathbb{P}_i) \tag{4.9}$$

其中 $\mathbb{P}_i$ 是 $\mathrm{sat}(\mathcal{A}_i)$ 的有限基.

§4.4 微分情形的零点分解算法

1. 微分多项式与微分升列. 设 $\mathbb{K}$ 是一个特征为零的域, 称 δ 是 $\mathbb{K}$ 上的一个微分算子, 如果对 $\mathbb{K}$ 中的任意元素 a, b 有 $\delta a \in \mathbb{K}$, 而且 $\delta(a+b) = \delta a + \delta b$, $\delta(ab) = b\delta a + a\delta b$. $\Delta = \{\delta_1, \cdots, \delta_m\}$ 是 $\mathbb{K}$ 中有限个微分算子的集合. 如果有 $\delta_i(\delta_j(a)) = \delta_j(\delta_i(a))$, 则称 $\mathbb{K}$ 为一个微分域.

设 $\mathbb{K}$ 是一个具有微分算子 $\Delta = \{\delta_1, \cdots, \delta_m\}$ 的微分域. 记

$$\Theta = \{\delta_1^{\alpha_1} \cdots \delta_m^{\alpha_m} | \alpha_i \in N^+, i = 1, \cdots, m\}$$

Θ 中的一个元素 $\theta = \delta_1^{a_1} \cdots \delta_m^{a_m}$ 的阶是一个整数 $\mathrm{ord}(\theta) = \sum_{i=1}^m a_i$. 对于 Θ 中的 $\theta = \delta_1^{a_1} \cdots \delta_m^{a_m}$, $\phi = \delta_1^{b_1} \cdots \delta_m^{b_m}$, 我们定义 $\theta \circ \phi = \delta_1^{a_1+b_1} \cdots \delta_m^{a_m+b_m}$.

设 $\mathbb{X} = \{x_1, \cdots, x_n\}$ 是微分未定元, $\Theta\mathbb{X} = \{\theta x_i | \theta \in \Theta, i = 1, \cdots, n\}$. $\mathbb{R} = \mathbb{K}\{x_1, \cdots, x_n\} = \mathbb{K}[\Theta\mathbb{X}]$ 表示未定元为 $x_1, \cdots, x_n$ 系数在 $\mathbb{K}$ 上的微分多项式环, 简记为 $\mathbb{K}\{\mathbb{X}\}$. 我们可以定义一个 $\Theta\mathbb{X}$ 中的一个全序 "$<$", 满足

1. $\forall u \in \Theta\mathbb{X}, \forall i, i = 1, \cdots, m, u < \delta_i u$,
2. $\forall u, v \in \Theta\mathbb{X}, \forall i, i = 1, \cdots, m, u < v \to \delta_i u < \delta_i v$.

在 MMP 系统中我们对 $\Theta\mathbb{X}$ 定义的全序为给定 $\theta_1 x$, $\theta_2 y$, $\theta_1 = \delta_1^{a_1} \cdots \delta_m^{a_m}$, $\theta_2 = \delta_1^{b_1} \cdots \delta_m^{b_m}$. 我们定义 $\theta_1 x < \theta_2 y$, 如果 $x < y$ 或者 $x = y$, $\mathrm{ord}(\theta_1) < \mathrm{ord}(\theta_2)$ 或者 $x = y$, $\mathrm{ord}(\theta_1) = \mathrm{ord}(\theta_2)$, 而序列 $[a_1, \cdots, a_m]$ 作为字典序小于序列 $[b_1, \cdots, b_m]$.

设 $\mathbb{K}$ 是一个微分域, $\mathbb{E}$ 是包括 $\mathbb{K}$ 的一个微分泛域. 对于 $\mathbb{K}\{x_1, \cdots, x_n\}$ 上的一个多项式 F, $\mathbb{E}^n$ 中的点 $\xi = (\xi_1, \cdots, \xi_n)$ 称为 F 的一个微分零点, 如果 $F(\xi_1, \cdots, \xi_n) = 0$. 设 $\mathbb{P} = \{P_1, \cdots, P_s\}$ 为 $\mathbb{K}\{\mathbb{X}\}$ 中的多项式集合, $\mathbb{E}^n$ 中的点 $\xi = (\xi_1, \cdots, \xi_n)$ 称为

$\mathbb{P}$ 的一个*微分零点*, 如果 $P_i(\xi)=0, i=1,\cdots,s$. $\mathbb{P}$ 的所有零点组成的集合称作 $\mathbb{P}$ 的*零点集*. 一般用 dZero($\mathbb{P}$) 表示.

设 $\mathbb{P}$ 是一个微分多项式集合, D 是一个微分多项式, 我们定义

$$\mathrm{dZero}(\mathbb{P}/D)=\{\xi\in\mathbb{E}^n|\xi\in\mathrm{dZero}(\mathbb{P}),D(\xi)\neq 0\}$$

设 $\mathbb{P},\mathbb{D}$ 是两个多项式集合, 我们定义

$$\mathrm{dZero}(\mathbb{P}/\mathbb{D})=\{\xi\in\mathbb{E}^n|\xi\in\mathrm{dZero}(\mathbb{P}),\text{对}D\in\mathbb{D},\text{都有}D(\xi)\neq 0.\}$$

设 $\mathbb{P}_i,\mathbb{D}_i$, $i=1,\cdots,m$ 是两组多项式集合, 我们称 $\cup_{i=1}^m\mathrm{dZero}(\mathbb{P}_i/\mathbb{D}_i)$ 为一个*微分拟代数簇*.

设 f 是 $\mathbb{K}\{\mathbb{X}\}$ 中的微分多项式, f 的*类*定义为最大的下标 p, 使得 x_p 或 x_p 的微分在多项式 f 中真正出现, 记为 cls(f). 如果 $f\in\mathbb{K}$, $\mathrm{cls}(f)=0$. x_p 称为该微分多项式的*主变元*. f 的*领式*是 $\Theta\mathbb{X}$ 中出现在 f 中的最大元, 记为 $v_f=\phi x_p$. f 可写作

$$f=I_d v_f^d+\cdots+I_0$$

$\mathrm{I}(f)=I_d$ 称为 f 的*初式*, $\mathrm{S}(f)=\dfrac{\partial f}{\partial v_f}$ 是 f 的 *隔离子*. 对于非零 $\theta\in\Theta$, 我们有

$$\theta f=\mathrm{S}(f)(\theta\circ\phi)x_p+\text{序低于}\theta\circ\phi x_p\text{的项}$$

设 g 是不在 $\mathbb{K}$ 中的一个微分多项式. 我们定义 $f<g$, 如果

1. $v_f<v_g$, 或者
2. $v_f=v_g=v$, $\deg(f,v)<\deg(g,v)$.

如果 $f<g$ 与 $g<f$ 都不成立, 我们就说 f 和 g 有相同的序. 我们称 f 对 g 是*部分约化的*, 如果 f 不含 v_g 的微分. f 对 g 是*微分约化的*, 简称为约化的, 如果 f 不含 v_g 的微分, 并且 $\deg(f,v_g)<\deg(g,v_g)$. f 称为对多项式集合 S 是 *(部分)* 约化的, 如果 f 对 S 中任何的元素是 (部分) 约化的.

设 $\mathcal{A}$ 是一个微分多项式集合, 如果 $\mathcal{A}$ 中的任意一个元素对 A 中任意其他的元素是约化的, 则称 $\mathcal{A}$ 是一个*自约化集*. 一个自约化集中的元素个数总是有限的. 将一个自约化集排序 $\mathcal{A}=A_1,\cdots,A_p$ 满足：$A_1<\cdots<A_p$, 称 $\mathcal{A}$ 是一个*升列*. 设 $\mathcal{A}$ 是一个微分升列, 我们分别用 $I_{\mathcal{A}}$ 和 $S_{\mathcal{A}}$ 表示 $\mathcal{A}$ 中的多项式的初式和隔离子的乘积.

对于升列 $\mathcal{A}=A_1,\cdots,A_p$ 和升列 $\mathcal{B}=B_1,\cdots,B_q$, 我们说 $\mathcal{A}$ 是小于 $\mathcal{B}$ 的, 如果存在 $j\leqslant\min(p,q)$ 使得对任意 $i<j$, A_i, B_i 有相同的序, 而 $A_j<B_j$, 或者 $q<p$ 并且对于 $i=1,\cdots,q$, A_i, B_i 有相同的序.

微分升列也满足以下基本性质.

引理 4.4.1　*序严格降低的微分升列序列是有限的.*

$\mathbb{K}\{x_1,\cdots,x_n\}$ 中的一个集合 F 中的升列一定有一个最小元, 称为 F 的基列. F 是一个微分多项式集合, f 对 F 的基列是约化的, 则 $F\cup\{f\}$ 的基列一定小于 F 的基列, 即

引理 4.4.2　*在一个微分多项式集合中加入一个对于其基列约化的微分多项式, 则新的多项式集合的基列的序比原集合的基列的序要低.*

$\mathbb{R}$ 中的一个理想 I 称作是一个*微分理想*, 如果满足：对任意的 $\delta\in\Delta$, 和 $f\in I$, $\delta f\in I$. 一个理想 I 称作是一个*完全理想*(微分根理想), 如果 I 是一个微分理想, 并且对任意的 $f\in R$, 如果 f 的某一个幂次属于 I 则一定有 f 属于 I. 如果 S 是 R 的一个子集, 我们分别用 $(S),[S],\{S\}$ 表示有 S 生成的代数意义下的理想, 微分理想, 完全微分理想. 微分多项式集合 S 称作是平凡的, 如果 $[S]$ 是 $\mathbb{K}\{x_1,\cdots,x_n\}$.

设 $\mathcal{A}=A_1,\cdots,A_p$ 是一个升列, A_i 和 A_j 满足 $v_{A_i}=\theta_i x_k, v_{A_j}=\theta_j x_k$. 设 θx_k 是 $\theta_i x_k$ 和 $\theta_j x_k$ 的最小的公共微分, 那么存在 $\phi_i\in\Theta$ 和 $\phi_j\in\Theta$ 使得 $\theta=\theta_i\circ\phi_i=\theta_j\circ\phi_j$. 我们定义 $\Delta_{i,j}=S(A_j)\phi_iA_i-S(A_i)\phi_jA_j$,

$$\Delta(\mathcal{A})=\{\Delta_{i,j}|1\leqslant i,j\leqslant p\}$$

$\mathcal{A}$ 称为是*可积*的, 如果 $\Delta(\mathcal{A})$ 为空或对 $\Delta(\mathcal{A})$ 中任意的 $\Delta_{i,j}$, 都存在 $\mathcal{A}$ 中的有限个初式和隔离子的乘积 H 使得 $H\Delta_{i,j}$ 属于理想 $(\{\phi g|\phi\in\Theta,g\in\mathcal{A},v_{\phi g}<\theta x_k\})$. 显然, 如果 $\Delta(\mathcal{A})$ 中的所有多项式对 $\mathcal{A}$ 的余式都是 0, 则 A 是可积的.

2. 微分多项式的表示.　在 MMP 系统中我们通过 depend 命令来定义微分多项式系统中的自变量和微分未定元. 命令 depend (DiffIndetList, IndepVarList) 申明 DiffIndetList 中的微分未定元是依赖 IndepVarList 中的独立变量的. 在 MMP 的一个过程中最好只使用一次 depend 命令. 否则 depend 命令会引起语义的混淆. 如果微分未定元是 $x_1,\cdots,x_n$, 微分域中的微分为 $\Delta=\{\frac{\partial}{\partial t_1},\cdots,\frac{\partial}{\partial t_m}\}$, 那么我们用 MMP 命令depend($[x_1,\cdots,x_n],[t_1,\cdots,t_m]$) 来确定 $x_1,\cdots,x_n$ 对 $t_1,\cdots,t_m$ 是依赖的, 而用命令 depend show() 来查询已经定义的微分依赖关系.

$\delta_1^{\alpha_1}\cdots\delta_m^{\alpha_m}x$ 即 $\frac{\partial x_1^{\alpha}+\cdots+\alpha_m}{\partial t_1^{\alpha_1}\cdots\partial t_m^{\alpha_m}}$ 在 MMP 系统中表示为 $x[\alpha_1,\cdots,\alpha_m]$. MMP 中的函数 diff(x, t, n)返回 $\frac{\partial^n x}{\partial t^n}$.

```
>depend([x, y, z], [u, v, w]);
1
>dependshow();
[[x, y, z], [u, v, w]]
>f:=diff(x, u, 2);
```

```
f:=x[2, 0, 0]
>g:=diff(f, v, 3);
g:=x[2, 3, 0]
>diff(y[1, 2, 3], u, 2);
y[3, 2, 3]
>diff(t, u, 1)
0
```

MMP 函数 leader来计算微分多项式的领式. MMP 函数 dleast来计算微分多项式组中的最小元素. MMP 函数 dbasicset来计算一个微分多项式组的基列.

```
>depend([x, y, r], [t]):
>p1:=m*x[2]+r*x:
>p2:=m*y[2]+r*y-g:
>p3:=x^2+y^2-l^2:
>ord:=[r, x, y]:
>G:=[p1, p2, p3];
G:=[m*x[2]+x*r, -g+m*y[2]+y*r, -l^2+y^2+x^2]
>leader(p1, [x, y, r]);
x[2]
>dleast(G, ord);
x^2+y^2-l^2
>dbasicset(G, ord);
[r*y+y[2]*m-g, x^2+y^2-l^2]
```

3. 微分特征列方法. 对于给定的微分多项式 F, G 和微分未定元 x, 与代数情形类似, 存在非负整数 s, t 满足下面的余式公式：

$$I^s S^t F = \sum_i P_i \phi_i G + R$$

其中 $I = \mathrm{I}(G)$, $S = \mathrm{S}(G)$, P_i 是微分多项式, $\phi_i G$ 是对 G 的某个微分, R 对微分多项式 G 关于 x 是约化的.

MMP 的函数 dprem计算两个微分多项式的余式.

```
>dprem(p1, p3, x);
8*r*y^4+8*y[2]*m*y^3-16*l^2*r*y^2-8*y[2]*l^2*m*y+8*l^4*r
-8*y[1]^2*l^2*m
>dprem(p2, p3, y);
```

Algorithm 19: dprem(F, G, x)

Input: 多项式 F, G.

Output: F 对 G 的关于变元 x 的余式.

$R = F$

$\phi x = G$ 的关于未定元 x 的领式

while R 对 F 关于变元 x 没有约化 **do**

  $\theta \circ \phi x = R$ 中关于 x 的最大的微分

  $R = \mathrm{prem}(R, \theta G, \theta \circ \phi x)$

end

return R

```
8*r*x^4+8*x[2]*m*x^3+8*g*y*x^2-16*l^2*r*x^2
-8*l^2*x[2]*m*x-8*l^2*g*y+8 *l^4*r-8*x[1]^2*l^2*m
```

对于一个微分多项式 f 与一个微分升列 $\mathcal{A} = A_1, \cdots, A_p$, 我们可以计算 f 对 $\mathcal{A}$ 的伪余式, $R = \mathrm{dpremas}(f, \mathcal{A})$, 满足下面的余式公式：

$$I_1^{i_1} \cdots I_p^{i_p} S_1^{s_1} \cdots S_p^{s_p} f = R \mod [A]$$

其中 $I_j = \mathrm{I}(A_j)$, $S_k = \mathrm{S}(A_k)$, i_j, s_k 是非负整数, R 对于 $\mathcal{A}$ 是约化的.

算法 dpremas 计算一个微分多项式对一个微分升列的余式.

Algorithm 20: dpremas(P, $\mathcal{A}$)

Input: 多项式 P, 自约化集 $\mathcal{A}$

Output: R, 为 P 对自约化集 $\mathcal{A}$ 的余式.

$R = P$

while R 对 $\mathcal{A}$ 没有约化**do**

  $F = \mathcal{A}$ 没有对 R 约化次序最高的多项式

  x=F 的主变元

  $R = \mathrm{dprem}(R, F, x)$

end

return R

MMP 中的函数 dpremas 计算微分多项式对微分升列的余式. MMP 中的函数 dremset 计算微分多项式集合对微分升列的余式.

```
>depend([u, v, w], [x, y]);
>F:=[u[0, 1]+1, -y+x*u[1, 0]-u, v[1, 0]-1, y*v[0, 1]+x-v, v[0, 1]
    -u[1, 0], v[1, 0]-w^2*u[0, 1]+w*v[0, 1]+w*u[1, 0]-v*w[0, 1]-
    u*w[1, 0]]:
```

```
>ord:=[u, v, w]:
>A:=dbasicset(F, ord);
A:=[-u+u[1, 0]*x-y, u[0, 1]+1, v[1, 0]-1, -v+x+v[0, 1]*y]
>dpremas(part(F, 5), A, ord);
-y*u+x*v-x^2-y^2
>dpremas(part(F, 6), A, ord);
y*w*u-w[1, 0]*y*x*u+x*w*v-w[0, 1]*y*x*v+y*x*w^2-x^2*w+y^2*w+y*x
>B:=dremset(F, A, ord);
B:=[-y*u+x*v-x^2-y^2, y*w*u-w[1, 0]*y*x*u+x*w*v-w[0, 1]*y*x*v
+y*x*w^2-x^2*w+y^2*w+y *x]
```

对于偏微分多项式系统, 我们要求微分升列满足可积性条件. 算法 dcharset 计算一个微分多项式系统的满足可积性条件的微分特征列.

算法:

Algorithm 21: dcharset($\mathcal{P}$)

Input: 多项式集合 $\mathbb{P}$

Output: 升列 $\mathcal{A}$: 为 $\mathbb{P}$ 的特征列.

repeat

  $\mathcal{A}$ =dbasicset($\mathbb{P}$)

  $\mathbb{R}$ ={ dpremas($q, \mathcal{A}$) $\neq 0 | q \in \mathcal{P}$}

  $\mathbb{D}$ = {dpremas($s, \mathcal{A}$) $\neq 0 | s \in \Delta(\mathcal{A})$}

  $\mathbb{P} = \mathcal{P} \cup \mathbb{R} \cup \mathbb{D}$

until $\mathbb{R} \cup \mathbb{D} \neq \emptyset$;

return $\mathcal{A}$

MMP 中的函数 dcharset 计算一个多项式集合的特征列.

```
>dcharset(F, [w, v, u]);
[2*y*w*u-w[1, 0]*y*x*u-w[0, 1]*y^2*u+y*x*w^2+2*y^2*w
-w[0, 1]*y*x^2+y*x-w[0, 1]*y^3,
-y*u+x*v-x^2-y^2, -u+u[1, 0]*x-y, u[0, 1]+1]
```

定理 4.4.3 设 F 是一个微分多项式组, 存在一个算法, 在有限步内得到有限个可积微分升列 A_i, 使得

$$\mathrm{dZero}(F) = \cup_i \mathrm{dZero}(\mathcal{A}_i/(\mathrm{I}_{\mathcal{A}_i}\mathrm{S}_{\mathcal{A}_i}))$$

MMP 系统中一个重要的函数dwsolve用来给出各种形式的零点分解. 下面我们来介绍一下 MMP 中 dwsolve 的调用方法.

```
dwsolve(F, Y)
dwsolve(F, Y, D)
dwsolve(F, Y, D, type)
```

参数: F – 变元为U,Y的微分多项式集合.

Y – 变元表(U中的变元将被作为参数处理).

D – 变元为U,Y的多项式的表, 可以是空表.

type – 只能取 "ritt", "wu", "weak" 之一. 缺省值为"ritt".

输出: 一个升列构成的表.

如果输出是 $[\mathcal{A}_1,\cdots,\mathcal{A}_r]$, 那么有

$$\mathrm{dZero}(F/D)=\cup_{i=1}^{r}\mathrm{dZero}(\mathcal{A}_i/I_{\mathcal{A}_i}S_{\mathcal{A}_i}D)$$

升列 $\mathcal{A}_i$ 具有参数 'type' 指定的形式. dwsolve 还采取了下面措施增加运算速度.

- 首先将 F 中的多项式在有理数域中因式分解而分成多个子多项式组.
- 对于每个子系统, 应用特征列序列方法, 并且将所有新产生的多项式都在有理数域上因式分解.
- 如果某个升列的所有正常零点是 D 的零点, 则可以删除这一升列.

§4.5 拟代数簇的投影运算

前面我们已经定义了拟代数簇的概念, 本节将介绍拟代数簇投影的概念和计算方法.

1. 代数情形拟代数簇的投影算法. 设 $m<n$, 我们定义 $\mathbb{E}^n$ 空间到 $\mathbb{E}^m$ 上的投影运算为

$$\begin{array}{rlcl}\mathrm{Proj}: & \mathbb{E}^n & \to & \mathbb{E}^m\\ & (a_1,\cdots,a_m,a_{m+1},\cdots,a_n) & & (a_1,\cdots,a_m)\end{array}$$

我们将投影运算应用到拟代数簇 $\Psi=\cup_{i=1}^{a}\mathrm{Zero}(\mathcal{A}_i/D_i)$ 上, 其在坐标 $x_1,\cdots,x_m$ 上的投影为

$$\mathrm{Proj}_{x_{m+1},\cdots,x_n}\Psi=\{a\in\mathbb{E}^m\mid\exists b\in\mathbb{E}^{(n-m)}\mathrm{s.t.}(a,b)\in\Psi\}$$

当 $m=0$ 时, 如果 $\Psi\neq\emptyset$, 定义 $\mathrm{Proj}_{x_1,\cdots,x_n}\Psi=1$; 否则为 0.

两个拟代数簇并集的投影等于分别投影后的并集, 但是两个拟代数簇交集的投影并不等于分别投影后的交集. 投影运算对代数簇是不封闭的, 即代数簇的投影有可能不是代数簇. 投影运算对拟代数簇是封闭的, 即拟代数簇的投影仍然是一个拟代数簇.

下面是关于计算投影的三个基本引理.

引理 4.5.1 对 $\mathbb{K}[\mathbb{X}]$ 中的任一多项式 $Q=\sum_{i=0}^{t}Q_ix_n^i$, 这里 Q_i 仅含有变元 $x_1,\cdots,x_{n-1}$,

$$\mathrm{Proj}_{x_n}\mathrm{Zero}(\emptyset/Q)=\cup_{i=0}^{t}\mathrm{Zero}(\emptyset/Q_i)$$

引理 4.5.2 令 P, Q 是 $\mathbb{K}[\mathbb{X}]$ 中的两个多项式, 且 $\deg(P,x_n)>0$, $\deg(Q,x_n)=0$, 那么

$$\mathrm{Proj}_{x_n}\mathrm{Zero}(P/\mathrm{I}(P)Q)=\mathrm{Zero}(\emptyset/\mathrm{I}(P)Q)$$

引理 4.5.3 令 P,Q 是 $\mathbb{K}[\mathbb{X}]$ 中的两个多项式, 且 $d=\deg(P,x_n)>0$, $\deg(Q,x_n)>0$.

$$\mathrm{Proj}_{x_n}\mathrm{Zero}(P/\mathrm{I}(P)Q)=\mathrm{Proj}_{x_n}\mathrm{Zero}(\emptyset/\mathrm{I}(P)R)$$

其中 $R=\mathrm{prem}(Q^d,P,x_n)$.

定理 4.5.4 存在一个算法在有限步内能计算出一个拟代数簇的投影, 而且拟代数簇的投影仍然是一个拟代数簇.

对于多项式集合 $\mathbb{P}$ 和多项式 D, 为了计算 $\mathrm{Zero}(\mathbb{P}/D)$ 到 $x_1,\cdots,x_m$ 上的投影, 首先, 对 $\mathbb{P}$ 在给定变元序 $x_1<\cdots<x_m<x_{m+1}<\cdots<x_n$ 下进行零点分解, 得到一组升列 $A_1,\cdots,\mathcal{A}_r$, 有如下的零点关系

$$\mathrm{Zero}(\mathbb{P}/D)=\cup_{i=1}^{r}\mathrm{Zero}(\mathcal{A}_i/I_{\mathcal{A}_i}D)$$

其次, 对于每一个 i, 我们利用上面提到的引理来计算 $\mathrm{Zero}(\mathcal{A}_i/I_{\mathcal{A}_i}D)$ 到 $x_1,\cdots,x_m$ 上的投影. 这里一个关键的事实是, 当我们计算 $\mathrm{Proj}_{x_n}\mathrm{Zero}(\mathcal{A}_i/I_{\mathcal{A}_i}D)$ 时, 得到的结果仍然是升列形式. 因而, 我们可以继续计算 $\mathrm{Proj}_{x_{n-1}}$. 最后, 由于拟代数簇并的投影等于拟代数簇投影的并, 从而得到拟代数簇 $\mathrm{Zero}(\mathbb{P}/D)$ 的投影.

对于升列 $\mathcal{C}:C_1,\cdots,C_s$, 我们可以给出计算 $\mathrm{Zero}(\mathcal{C}/I_{\mathcal{C}}D)$ 的投影的算法 projectas.

对于给定的升列 $\mathcal{C}$ 和多项式 D 算法 ComplectProjectionCS 可以判定 $\mathrm{Zero}(\mathcal{C}/I_{\mathcal{C}}D)$ 是否为空.

综合算法 projectas 和 ComplectProjectionCS, 对于多项式系统 $\mathbb{P}$ 和多项式 D, 算法 project 可以得到 $\mathrm{Zero}(\mathbb{P}/D)$ 的投影.

对于一般的拟代数簇 $\cup_i\mathrm{Zero}(\mathbb{P}_i/D_i)$, 可以先得到 $\mathrm{Zero}(\mathbb{P}_i/D_i)$ 的投影, 再把这些投影并起来就是拟代数簇 $\cup_i\mathrm{Zero}(\mathbb{P}_i/D_i)$ 的投影.

MMP 的函数 proj(ps, ds, ord, pord) 可以用来计算多项式的投影, 其中 ps 与 ds 为多项式集合, ord 与 pord 为变量集合. proj 计算 $\mathrm{Proj}_{\mathrm{pord}}\mathrm{Zero}(\mathrm{ps}/\mathrm{ds})$. 下面是 MMP 的关于拟代数簇投影计算的例子.

Algorithm 22: projectas: $(\mathcal{C}, D, \{x_{m+1}, \cdots, x_n\})$

Input: $\mathcal{C}: C_1, \cdots, C_r$: 是变量序 $x_1 < \cdots < x_n$ 下的一个升列

D: 一个多项式

Output: 一组 (B_i, D_i), B_i 是升列, D_i 是多项式, 使得

$$Proj_{x_{m+1},\cdots,x_n}\mathrm{Zero}(\mathcal{C}/I_{\mathcal{C}}D) = \cup_i \mathrm{Zero}(B_i/D_i)$$

if $\mathrm{cls}(C_r) \leqslant m$ 而且 $\mathrm{cls}(D) \leqslant m$ **then return** (C, D)

if $\mathrm{cls}(C_r) \leqslant m$ 而且 $\mathrm{cls}(D) > m$ **then**

  将多项式 D 写为关于 $x_{m+1}, \cdots, x_n$ 的多项式, 系数为 $D_{11}, \cdots, D_{1t_1}$

  return $(C, I_{\mathcal{C}}D_{11}) \cup \cdots \cup (C, I_{\mathcal{C}}D_{1t_1})$

end

$c_1 = \mathrm{cls}(C_r)$

$c_2 = \mathrm{cls}(D)$ (显然有 $c_1 > m$)

if $c_1 > c_2$ **then**

  $C' = C_1, \cdots, C_{r-1}$

  return projectas $(C', I(C_r)D, \{x_{m+1}, \cdots, x_n\})$

else if $c_1 < c_2$ **then**

  将 D 写为关于 x_{c_2} 的多项式, 系数为 $D_{21}, \cdots, D_{2t_2}$,

  return projectas$(C, D_{21}, \{x_{m+1}, \cdots, x_n\}) \cup \cdots \cup$

  projectas$(C, D_{2t_2}, \{x_{m+1}, \cdots, x_n\})$

else

  $C' = C_1, \cdots, C_{r-1}$

  $c = c_1$ $(c_1 = c_2)$

  $d = \deg(C_r, x_c)$

  $D' = \mathrm{prem}(D^d, C_r, x_c)$

  将 D' 写成关于 x_c 的多项式, 系数为 $D_{31}, \cdots, D_{3t_3}$,

  return projectas$(C', I(C_r)D_{31}, \{x_{m+1}, \cdots, x_n\}) \cup \cdots \cup$

  projectas$(C', I(C_r)D_{3t_3}, \{x_{m+1}, \cdots, x_n\})$

end

Algorithm 23: 算法：ComplectProjectionCS$(\mathcal{C}, D)$

Input: $\mathcal{C}$：一升列, D：一个多项式.

Output: 1 如果 $\mathrm{Zero}(\mathcal{C}/I_{\mathcal{C}}D)$ 非空, 0 如果 $\mathrm{Zero}(\mathcal{C}/I_{\mathcal{C}}D)$ 是空集.

A=projectas$(\mathcal{C}, D, \{x_1, \cdots, x_n\})$(假设$A = \cup_{i=1}^{r}(\emptyset, D_i)$)

if 存在一个 D_i 为非 0 常数 **then return** 1 **else return** 0

Algorithm 24: project($\mathbb{P}, D, \{x_{m+1}, \cdots, x_n\}$)

Input: $\mathbb{P}$: 一组多项式, D: 一个多项式

Output: 一组 (C_i, D_i) 使得 $Proj_{x_{m+1},\cdots,x_n}\mathrm{Zero}(\mathbb{P}/D) = \mathrm{Zero}(C_i/D_i)$

$A = \mathrm{wsolve}(\mathbb{P}, \{x_1, \cdots, x_m\})$

$B = \cup_{\mathcal{A}\in A}\mathrm{projectas}(\mathcal{A}, D, \{x_{m+1}, \cdots, x_n\}) = \cup_j \mathrm{Zero}(C_j/\mathrm{I}(C_j)D_j)$

$C = \emptyset$

foreach j **do**

  if ComplectProjectionCS$(C_j, D_j) = 1$ **then**

    $C = C \cup \{(C_j, D_j)\}$

  end

end

return C

```
>ps:=[x3*(x5^2-(x4-x1)^2)-2*x1*x5*(x4-x1),
     x3*(x5^2-(x4-x2)^2)-2*x2*x5*(x4-x2)];
>ds:=[(x1*(x5-x3)+x3*x4)*(x3*(x5-x3)-x2*x4)
     +(x2*(x5-x3)+x3*x4)*(x3*(x5-x3)-x1*x4)];
>ord:=[x5, x4, x3, x2, x1];
>pord:=[x5, x4];
>proj(ps, ds, ord, pord);
The result of projection is:
[[[x1-x2], [x3, x3^2+x1^2]], [[x1-x2, x3], [x1]]]
>ps:=[2*u1-x2*x5+x3*x4, 2*u2+x5*x1, 2*u3-x1*x3,
     2*u4-(x2-x1)*x5+x3*x4-x1*x3];
>ds:=[u1];
>ord:=[x5, x4, x3, x2, x1, u4, u3, u2, u1];
>pord:=[x5, x4, x3, x2, x1];
>proj(ps,ds,ord,pord);
The result of projection is:
[[[u1+u2+u3-u4], [u1, u3]],
 [[u2, u3, u1+u2+u3-u4], [u1]], [[u3, u1+u2+u3-u4], [u1, u2]]]
```

由上述算法给出的结果一般包含很多的分支. 我们可以将这些分支进行化简. 具体过程这里不再叙述.

2. 微分情形的拟代数簇投影算法. 与代数情形类似, 我们可以定义微分拟代数簇以及微分拟代数簇的投影运算. 设 $\mathbb{P}$ 是 $\mathbb{K}\{\mathbb{X}\}$ 中的一个微分多项式集合, D 是 $\mathbb{K}\{\mathbb{X}\}$

中的一个微分多项式, 我们定义:

$$\mathrm{dproj}_{x_{m+1},\cdots,x_n}\mathrm{dZero}(\mathbb{P}/D)=\{a\in\mathbb{E}^m\mid\exists b\in\mathbb{E}^{(n-m)}\mathrm{s.t.}(a,b)\in\mathrm{dZero}(\mathbb{P}/D)\}$$

当 $m=0$ 时, 如果 $\mathrm{dZero}(\mathbb{P}/D)\neq\emptyset$, 定义 $\mathrm{dproj}_{x_1,\cdots,x_n}\mathrm{dZero}(\mathbb{P}/D)=1$, 否则定义 $\mathrm{dproj}_{x_1,\cdots,x_n}\mathrm{dZero}(\mathbb{P}/D)=0$.

微分拟代数簇投影运算的关键是下列结果 ([81]).

引理 4.5.5 设 $\mathcal{A}\subset\mathbb{K}\{x_1,\cdots,x_n\}$ 是一个可积的微分升列, D 是一个对 $\mathcal{A}$ 部分约化的微分多项式, 则 $\mathcal{A}=0, I_{\mathcal{A}}S_{\mathcal{A}}D\neq 0$ 有微分零点的充分必要条件是它作为一个代数系统有代数零点.

设 $\mathcal{A}$ 是一个微分升列, D 是一个微分多项式, $R=\mathrm{dprem}(D,\mathcal{A})$, x 是一个微分未定元, 因为

$$\mathrm{dZero}(\mathcal{A}/I_{\mathcal{A}}S_{\mathcal{A}}D)=\mathrm{dZero}(\mathcal{A}/I_{\mathcal{A}}S_{\mathcal{A}}R)$$

所以自然就有

$$\mathrm{dproj}_x\mathrm{dZero}(\mathcal{A}/I_{\mathcal{A}}S_{\mathcal{A}}D)=\mathrm{dproj}_x\mathrm{dZero}(\mathcal{A}/I_{\mathcal{A}}S_{\mathcal{A}}R)$$

要特别注意的是, 在代数情形下, 投影将保持 D 的部分约化的性质, 同时容易知道, $I_{\mathcal{A}}$ 对 $\mathcal{A}$ 是部分约化的. 借助上述引理与吴零点分解定理, 我们可以将一般微分拟代数簇的投影化为代数情形的投影.

引理 4.5.6 设 $\mathcal{A}\subset\mathbb{K}\{x_1,\cdots,x_n\}$ 是一个可积的微分升列, D 是一个对 $\mathcal{A}$ 部分约化的微分多项式, $\mathbb{Y}$ 是 $\mathbb{X}$ 的一个子集, $\Theta\mathbb{Y}$ 是出现在 $\mathcal{A}$ 和 D 中的所有的关于 $\mathbb{Y}$ 的微分, 设 $\mathrm{Proj}_{\Theta\mathbb{Y}}\mathrm{Zero}(\mathcal{A}/I_{\mathcal{A}}S_{\mathcal{A}}D)=\cup_{i=1}^m\mathrm{Zero}(\mathcal{A}_i/I_{\mathcal{A}_i}D_i)$, 那么

$$\mathrm{dproj}_{\mathbb{Y}}\mathrm{dZero}(\mathcal{A}/I_{\mathcal{A}}S_{\mathcal{A}}D)=\cup_{i=1}^m\mathrm{dZero}(\mathcal{A}_i/I_{\mathcal{A}_i}D_i)$$

用下面的例子说明上述结果.

$$\begin{array}{ll}
\mathrm{dproj}_{x_1}\mathrm{dZero}(x_1''^2+x_2/x_3x_1'-x_4) & \exists x_1(x_1''^2+x_2=0\cap x_3x_1'-x_4\neq 0)\\
=\mathrm{Proj}_{x_1''}\mathrm{Zero}(x_1''^2+x_2/x_3x_1'-x_4) & \exists x_1''(x_1''^2+x_2=0\cap x_3x_1'-x_4\neq 0)\\
=\mathrm{Proj}_{x_1'}\mathrm{Zero}(/x_3x_1'-x_4) & \exists x_1'(x_3x_1'-x_4\neq 0)\\
=\mathrm{Zero}(/x_3)\cup\mathrm{Zero}(x_3/x_4) & \exists(x_3\neq 0\vee(x_3=0\cap x_4\neq 0))
\end{array}$$

由上述结果可以得到微分情形的投影定理.

定理 4.5.7 存在一个算法在有限步内能计算出一个微分拟代数簇的投影, 而且微分拟代数簇的投影仍然是一个微分拟代数簇.

给定开列 $\mathcal{A}$ 和一个多项式 D, 要计算 $\text{dproj}_{x_{m+1},\cdots,x_n}\text{dZero}(\mathcal{A}/D)$. 见算法 dprojas.

Algorithm 25: dprojas$(\mathcal{A}, D, \{x_{m+1}, \cdots, x_n\})$

Input: $\mathcal{A} : A_1, \cdots, A_r$ 一个微分可积升列

D: 一个微分多项式

Output: 一组 (B_i, D_i), 使得 $\text{dproj}_{x_{m+1},\cdots,x_n}\text{dZero}(\mathcal{A}/I_{\mathcal{A}}S_{\mathcal{A}}D) = \cup_i \text{dZero}(B_i/D_i)$

$R = \text{dpremas}(D, \mathcal{A})$

$\Theta\mathbb{X} = \{\theta x_i | i > m, \theta x_i$出现在$\mathcal{A}$或$R$中$\}$

return $\text{projectas}_{\Theta\mathbb{X}}(\mathcal{A}, R)$

给定多项式系统 $\mathbb{P}$ 和一个多项式 D, 要计算 $\text{dproj}_{x_{m+1},\cdots,x_n}\text{dZero}(\mathbb{P}/D)$, 见算法 dproj.

Algorithm 26: dproj$(\mathbb{P}, D)$

Input: $\mathbb{P}$ 一组微分多项式 D：一个微分多项式

Output: 一组 (B_i, D_i), 使得

$\text{dproj}_{x_{m+1},\cdots,x_m}\text{dZero}(\mathbb{P}/D) = \text{dZero}(B_i/D_i)$

对 $\mathbb{P}$, 进行零点分解得到一组可积的微分升列 $\mathcal{A}_1, \cdots, \mathcal{A}_r$, 有

$\text{dZero}(\mathbb{P}/D) = \cup_{i=1}^r \text{dZero}(\mathcal{A}_i/I_{\mathcal{A}_i}S_{\mathcal{A}_i}D)$

$R_i = \text{dpremas}(D, \mathcal{A}_i)$

return $\cup_{i=1}^r \text{dprojas}(\mathcal{A}_i, R_i, \{x_{m+1}, \cdots, x_n\})$

```
>depend([x, y], [t]);
>ps:=[x[1]-u*x^2-u^2*x, y-x^2];
>proj(ps, [], [x, y, u], [x], "diff");
The result of projection is:
[[[4*u^2*y^3-4*u^4*y^2+4*y[1]*u^2*y-y[1]^2],
[y, u, 2*u^2*y-y[1]]], [[y], [1]],
[[u, y[1]], [y]]]
>depend([x, y], [t, v]);
>ps:=[x[1, 0]^2+x+1, x[1, 1]^2*y[1, 0], x[0, 1]^2-y^2];
>proj(ps, [], [y, x], [y], "diff");
The result of projection is:
[[[x[0, 1], x+x[1, 0]^2+1], [x[1, 0]]]]
```

考虑微分拟代数簇投影法应用的几个例子.

例 4.5.8 考虑如下的控制系统：u 是控制参数或输入参数，变元 x 和输出变元 y，有下列关系

$$x' = ux^2 + u^2x; \qquad y = x^2$$

需要消去变元 x. 设 $\mathbb{P} = \{x' - ux^2 - u^2x, y - x^2\}$，首先由零点分解定理：$\mathrm{dZero}(\mathcal{P}) = \mathrm{dZero}(y'^2 - 4u^2yy' - 4u^2y^3 + 4u^4y^2, 2uyx - y' + 2u^2y/u(y' - 2u^2y)) \cup \mathrm{dZero}(u, y', x^2 - y/x) \cup \mathrm{dZero}(y, x)$. 我们有 $\mathrm{dproj}_x\mathrm{dZero}(\mathbb{P}) = \mathrm{dZero}(y'^2 - 4u^2yy' - 4u^2y^3 + 4u^4y^2/u(y' - 2u^2y)) \cup \mathrm{dZero}(u, y'/y) \cup \mathrm{dZero}(y)$.

例 4.5.9 $P_1 = 0$, $P_2 = 0$, $Q \neq 0$, 其中 $P_1 = \frac{\partial y_1}{\partial x_1} + 2y_2 + y_1^2$; $P_2 = -\frac{\partial y_1}{\partial x_2} + 4y_2 * y_1^2 - 4y_1\frac{\partial y_2}{\partial x_1} + 2\frac{\partial^2 y_2}{\partial x_1^2} + 8y_2^2$; $Q = \frac{\partial y_1}{\partial x_1}\frac{\partial y_2}{\partial x_2} + \left(\frac{\partial y_1}{\partial x_2}\right)^2$, 其中 y_1 是微分参数. 设 $y_1 < y_2$, 由零点分解定理，有

$$\mathrm{dZero}(\{P_1, P_2\}/Q) = \mathrm{dZero}(\{H, P_1\}/Q)$$

其中 $H = \frac{\partial y_1}{\partial x_2} + \frac{\partial^3 y_1}{\partial x_1^3} - 6y_1^2\frac{\partial y_1}{\partial x_1}$.

根据微分情形的投影算法，我们先计算 Q 对 H, P_1 的余式

$$R = 2\Big(\frac{\partial y_1}{\partial x_2}\Big)^2 - \frac{\partial y_1}{\partial x_1}\frac{\partial^2 y_1}{\partial x_2 \partial x_1} - 2\frac{\partial y_1}{\partial x_1}y_1\frac{\partial y_1}{\partial x_2}$$

因此 $\mathrm{dproj}_{y_2}\mathrm{dZero}(\{P_1, P_2\}/Q) = \mathrm{dproj}_{y_2}\mathrm{dZero}(\{H, P_1\}/R) = \mathrm{dZero}(H/R)$.

文献说明

吴文俊提出了多项式集合特征列的概念，给出了计算多项式组特征列的算法，并将特征列方法应用于几何定理自动证明中 ([85] 和 [86]).

MMP 中实现的基本零点分解算法是王定康根据 [86] 中的原始算法给出的改进算法 ([77]). MMP 中实现的基于吴升列的算法源于吴的论文 [101]. MMP 中实现的基于弱升列的算法源于周、高的论文 [23]. 正则升列由杨路，张景中，Kalkbrener 提出 ([104] 和 [54]). 其中关键引理 4.3.7 使用的是王论文中的算法 ([76]). MMP 中实现的基于正规升列的算法源于论文 [37]. 饱和升列由 Lazard 最先提出 ([60]).

将多项式系统分解为不可约升列的相关算法请参考 [86]，其中在代数扩域上因式分解的算法来自于袁的论文 [107].

关于微分情形的零点分解的相关算法，请参考 [72],[94], [88], [91], [17], [18], [20], [6] 和 [51]. MMP 使用的算法基于 [5]. 有关代数情形拟代数簇投影的计算可参考 [97],[37], [78] 和 [13]，微分情形微分拟代数簇的投影计算请参考 [40] 和 [81].

第五章　几何定理机器证明与发现

几何定理机器证明是吴特征列方法最成功的应用. 在吴文俊荣获 “Herbrand 自动推理杰出成就奖” 的颁奖词中对于几何定理机器证明的吴方法有如下评论：“吴的工作将几何定理证明从自动推理的一个不太成功的领域变为最成功的领域之一. 在很少的领域中, 我们可以讲机器证明优于人的证明. 几何定理证明就是这样的一个领域. ” 本章将介绍怎样在 MMP 中用吴方法自动证明几何定理, 自动发现几何公式.

§5.1　几何命题的输入与转换

MMP 接受四种几何命题的输入：自然语言形式, 构造形式, 谓词形式与代数形式. 这四种形式所能描述的几何命题的范围依次增加. 分别介绍如下.

1. 几何命题的自然语言形式.　考虑下面几何定理：

例 5.1.1　*$ABCD$ 是一平行四边形. 证明其两对角线 AC, BD 互相平分. 设 O 是 AC 与 BD 的交点. 证明 O 是 AC 的中点* (图 5.1).

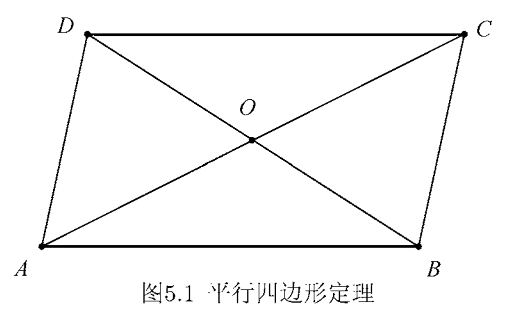


图5.1 平行四边形定理

为了方便使用, 我们为用户提供了一种接近 *自然语言* 的输入形式. 例如, 在 MMP 中可以如下证明上面的几何定理.

```
>wprove("Example Parallelogram. Let ABCD be a parallelogram.
O is the intersection of the diagonals AC and BD. Show that O is
the midpoint of AC. ");
The geometric Statement is valid!
1 components checked.
Time is: 141 microsecond.
```

这里, wprove 是 MMP 的一个函数, 它使用吴方法证明几何定理. wprove 的详细描述在下一节给出.

应该说明, 这里的 “自然语言” 实际上是一种严格定义的几何语言. 其完整定义

请见附录 A. 下面简单介绍其基本性质. 一个几何命题由三部分组成：命题名称, 几何假设与结论.

- 命题名称部分由 “Example” 开始, 后面跟一个单词, 是几何命题的名称. 也可以省略几何命题的名称.
- 假设部分由若干几何语句构成. 每个几何语句构造一个或多个几何元素. 下面是一些常用的几何语句.

```
M is the midpoint of P1P2
O is the circumcenter of triangle P1P2P3
P is a point on the circumscribed circle O of triangle ABC.
T is the intersection of line AB and the circle OA
A, B, C, and D are four points on a circle.
Y is the intersection of circle OA and the
      circumcircle of triangle ABC.
```

  MMP 不检查一个语句是否符合英文语法. 有些词, 例如 a, and, of with, the, another, to 在解释过程中将被省略. 在上面给出的语句中, 最后两个语句分别与下面语句等价.

```
A B C D are four points on circle.
Y is intersection of circle OA circumcircle of triangle ABC.
```

- 几何命题的结论由 “Show” 开始, 可以是以下命题：

```
Show that points P1, P2, and P3 are collinear.
Show that points P1, P2, P3, and P4 are cocircle.
Show that point P1 is the midpoint of  P2P3.
Show that [line] P1P2 is parallel to [line] P3P4.
Show that [line] P1P2 is perpendicular to [line] P3P4.
Show that [segment] P1P2 is congruent to [segment]  P3P4.
Show that angle P1P2P3 is congruent to angle  P4P5P6.
```

  也可以是一些简单的代数表达式：

```
P1P2=P3P4
P1P2*P3P4 = P5P6*P7P8
```

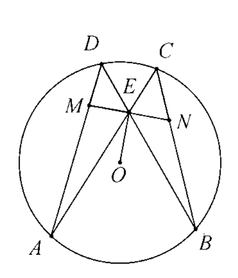
所以, 一个几何命题具有以下形式：

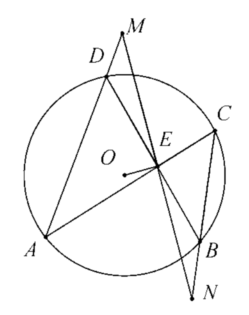
```
Example [name]. S1 ... Sk. CONC.
```

其中 Si 是一系列几何语句, 它们是几何命题的假设部分. CONC 是几何命题的结论. 下面再举几个例子.

例 5.1.2 (蝴蝶定理) 设 A, B, C, D 为圆上四点. $E = AC \cap BD$. 过点 E 与 EO 垂直的直线分别交 AD, BC 与 M, N. 证明 E 是 MN 的中点.

机器证明吴方法的一个特点是这一方法在证明时不考虑定理中几何元素的顺序. 所以用吴方法证明的几何定理实际上与定理的几何图形无关. 一个推论是, 一旦定理正确, 则对于所有的图形都正确. 例如, 蝴蝶定理对于图 5.2 中的三种情形都正确.





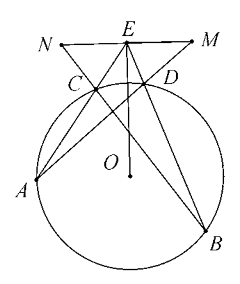


图 5.2 蝴蝶定理的三种情形

```
>wprove("Example Butterfly. Let A, B, C, D be four points on a
circle O. E is the intersection of line AC and line BD. M is the
intersection of line AD and the line passing through E and
perpendicular to OE. N is the intersection of line BC and line EM.
Show that E is the midpoint of NM. ");
```

例 5.1.3 (Pappus 定理) 设 A, B, C 与 A_1, B_1, C_1 分别为两直线上三点. $P = AB_1 \cap B_1A, Q = AC_1 \cap CA_1, R = BC_1 \cap CB_1$. 证明 P、Q、R 三点共线 (图 5.3).

```
>wprove("Example Pappus. Let A, B, and C be three points on a line;
and A1, B1, and C1, be three points on another line.
P is the intersection of line AB1 and line A1B. Q is the intersection
of lines AC1 and CA1. R is the intersection of lines BC1 and CB1.
Show that P, Q, and R are collinear.");
```

例 5.1.4 (Pascal 定理) 设 A, B, C, A_1, B_1, C_1 为圆上六点. $P = AB_1 \cap BA_1, Q = AC_1 \cap A_1C, R = BC_1 \cap C_1B$. 证明 P、Q、R 三点共线 (图 5.4).

```
>wprove("Example Pascal. Let A, B, C, A1, B1, and C1 be
six points on a circle. P is the intersection of line AB1 and line
BA1. Q is the intersection of line AC1 and line CA1. R is the
intersection of line BC1 and line CB1. Show that points P, Q, and
```

```
R are collinear.");
```

例 5.1.5 (Feuerbach 定理) 三角形 ABC 的内切圆与其九点圆相切 (图 5.5).

```
>wprove("Example Feuerbach. Show that the inscribed circle of the
triangle ABC is tangent with the nine point circle of the same
triangle.");
```

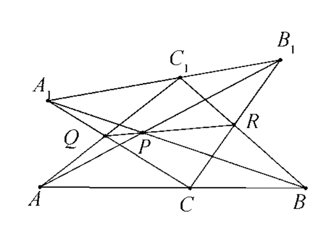


图5.3 Pappus定理

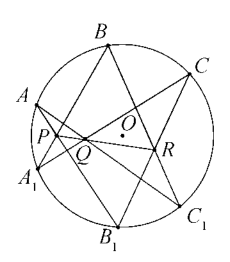


图5.4 Pascal定理

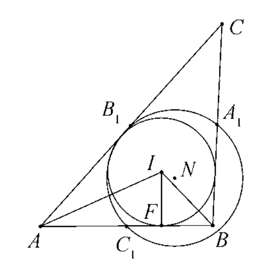


图5.5 Feuerbach定理

2. 几何命题的构造形式. 几何命题的另外一种描述方式是逐个构造几何命题中的几何元素. 如果在构造中只允许使用直线与圆, 则这样生成的几何图形就是可 尺规作图 的图形, 相应的几何命体称为 构造型几何命题. 例如, 例 5.1.1可以构造性描述如下：

```
[[[POINT, A, B, C],
[INTER, D, [PLINE, A, B, C], [PLINE, C, A, B]],
[INTER, O, [LINE, A, C], [LINE, B, D]]],
[[MIDPOINT, O, A, C]]];
```

解释如下：第一个作图语句作出三个点 A、B、C. 第二个作图语句作出一个点 D. 这个点是通过点 C 而平行于直线 AB 的直线 [pline, C, A, B] 与通过点 A 而平行于直线 BC 的直线 [pline, A, B, C] 的交点. 第三个作图语句作出直线 AC 与直线 BD 的交点 O. 命题的结论是 O 是线段 AC 的中点.

以上形式与前面所讲的自然语言描述是完全等价的. 使用 MMP 的 “geomtocs” 命令可以由一个几何命题的自然语言描述自动得到其构造性描述.

```
>geomtocs("Example Parallelogram. Let ABCD be a parallelogram.
O is the intersection of the diagonals AC and BD. Show that O is
the midpoint of AC. ");
[[[POINT, A, B, C], [INTER, D, [PLINE, A, B, C], [PLINE, C, A, B]],
[INTER, O, [LINE, A, C], [LINE, B, D]]], [[MIDPOINT, O, A, C]]];
```

一般讲, 一个构造型几何命题形如 [cs, conc], 其中 cs 是一个几何构造的列表, 称

为 *构造序列*. 任何一个构造序列都必须以作图命令 “point(自由点)” 或 “circle(圆)” 开始. 一个几何构造即用若干已经知道的几何元素构造一个新的几何元素. 我们这里采用了以点为基础的表示方式, 即一切几何元素都被表示为点的函数. 下面是一些典型的几何构造：

```
[circle, O, A, B, C];
[on, P, [line, A, B]];
[on, P, [cir, O, A]];
[inter, P, [tline, A, B, C], [cir, Q, A]];
```

以上四个几何构造的意义如下：作出以 O 为圆心的圆上的三个点 A, B, C; 在通过点 A 和 B 的直线 [line, A, B] 上取一个点 P; 在以 O 为圆心且通过点 A 的圆 [cir, O, A] 上取一个点 P; 作出通过点 A 而垂直于直线 BC 的直线 [tline, A, B, C] 与以 Q 为圆心且通过点 A 的圆 [cir, Q, A] 的交点. 几何命题的结论可以是以下几何关系：

```
[coll,A,B,C];[PARA,A,B,P,Q];[perp,A,B,P,Q];[cong,A,B,P,Q].
```

当用户的目的只是想画一个图, 而不想证明任何结论时, 可以将命题的结论置为 “[ ]”.

使用 geomtocs, Pappus 定理 (5.1.3) 与 Feuerbach 定理 (5.1.5) 的构造形式如下：

```
[[[POINT,A,B,A1,B1], [ON,C,[LINE,A,B]], [ON,C1,[LINE,A1,B1]],
[INTER, P, [LINE, A, B1], [LINE, A1, B]],
[INTER, Q, [LINE, A, C1], [LINE, C, A1]],
[INTER, R, [LINE, B, C1], [LINE, C, B1]]], [[COLL, P, Q, R]]];


[[[POINT,A,B,up1],[INCENTER,C,A,B,up1],[FOOT,up2,up1,A,B],
[MIDPOINT,up4,A,B],[MIDPOINT,up5,A,C],[MIDPOINT,up6,B,C],
[CIRCUMCENTER,up3,up4,up5,up6]],[[TANGENT,up1,up2,up3,up4]]];
```

在 Feuerbach 定理的构造性描述中, 引入了 up1, up2, up3 等辅助点.

3. 几何命题的谓词形式. 我们通常遇到的大部分几何命题是构造型几何命题. 例如, 专著 [15] 中所收录的 512 条几何定理约有 80% 是构造型几何命题. 而专著 [24] 中所收录的 400 条几何定理则全部是构造型的. 一般讲, 一个几何命题并不一定可以表示为构造形式. 为此, 我们引入几何命题的 *谓词形式*. 一个几何命题的谓词形式如下：

```
Stat = [pvs, mvs, pts, ps, ds, conc]
```

其中

- pvs 是命题中的非自由变量表. 变量的次序是由大到小.
- mvs 是命题中的自由变量表. 变量的次序是由大到小.
- pts 是命题中点的表列. 每个点后面的元素是其坐标. 坐标的元素必须是在 pvs 与 mvs 中定义的变量.
- ps 是命题中的等式型的几何关系或几何谓词.
- ds 是命题中的非等式型的几何关系, 又称为非退化条件.
- conc 是命题的结论.

称以上几何命题是正确的是指：对于所有的变量

$$\mathrm{ps}\wedge\mathrm{ds}\Longrightarrow\mathrm{conc}$$

例 5.1.1 的谓词表示如下：

```
[[y_O, x_O, y_D, x_D], [v_C, u_C, v_B],
[A, [0,0], B, [0,v_B], C, [u_C,v_C], D, [x_D,y_D], O, [x_O,y_O]],
[[para,D,A,B,C], [para,D,C,A,B], [coll,O,A,C], [coll,O,B,D]],
[[coll, A, B, C]], [[midpoint, O, A, C]]];
```

其意义如下：y_O, x_O, y_D, x_D 是主变量, v_C, u_C, v_B 是三个参变量. 点 A 的坐标是 (0, 0), 点 B 的坐标是 $(0, v_B)$, 点 C 的坐标是 (u_C, v_C), 点 D 的坐标是 (x_D, y_D), 点 O 的坐标是 (x_O, y_O). 几何关系 [para, D, A, B, C] 是指：DA 平行于 BC. 几何关系 [coll, O, A, C] 是指：O, A, C 三点共线. 几何关系 [coll, A, B, C] 是非退化条件, 即假定 A, B, C 三点不共线. 几何关系 [midpoint, O, A, C] 是结论, 表示 O 是 AC 的中点.

所谓谓词, 实际上就是几何关系. MMP 中常见的几何谓词包括：

```
[coll, A, B, C]                是指: A, B, C三点共线.
[para, A, B, P, Q]             是指: AB平行于PQ.
[perp, A, B, P, Q]             是指: AB垂直于PQ.
[cong, A, B, P, Q]             是指: AB=PQ.
[acong, A, B, C, P, Q, R]      是指: 角ABC = 角PQR.
[cir, A, B, C, D]              是指: A, B, C, D四点共圆.
[dis, A, B, d]                 是指: |AB| = d.
[area, A, B, C, k]             是指: 三角形ABC的有向面积为k.
```

MMP 的函数 geomtopd(gs) 可以将一个用自然语言或构造形式表示的几何定理转变为谓词形式. 例如, 下面的命令给出例 5.1.1 的谓词表示.

```
>geomtopd([[[POINT, A, B, C],
[INTER, D, [PLINE, A, B, C], [PLINE, C, A, B]],
```

```
[INTER, O, [LINE, A, C], [LINE, B, D]]], [[MIDPOINT, O, A, C]]]);
[[y_O, x_O, y_D, x_D], [v_C, u_C, v_B, u_B, v_A, u_A],
[A, [0,0], B, [0,v_B], C, [u_C,v_C], D, [x_D,y_D], O, [x_O,y_O]],
[[point, A, B, C], [para, D, A, B, C], [para, D, C, A, B],
[coll, O, A, C], [coll, O, B, D]], [[para, B, C, A, B],
[para, A, C, B, D]], [[MIDPOINT, O, A, C]]];
```

注意, MMP 生成的谓词形式与前面我们给出的谓词形式有所不同. 主要在于非退化条件. 对于构造型几何命题, MMP 可以自动生成几何非退化条件, 以使得每个几何构造可以严格地确定相应的几何元素. 具体讲, 每个几何构造在变为谓词形式时都生成两类谓词：等式型谓词与非等式型谓词, 其中非等式型谓词又称为非退化条件. 等式型谓词的生成是显然的. 下面具体介绍怎样由几何构造生成非退化条件.

- 对于几何构造 [on, o], 我们假定几何元素 o 是有意义的. 例如, 如果 $o =$[line, A, B], 则非退化条件是 $A \neq B$. 如果 $o =$[CIR, O, A], 则非退化条件是 $O \neq A$.
- 对于几何构造 [inter, o_1, o_2], 我们假定几何元素 o_1 与 o_2 存在交点. 例如, 如果 $o_1 =$[line,A, B], $o_2 =$[line,C, D], 则非退化条件是 $AB \nparallel CD$. 如果 $o_1 =$[line,A, B], $o_2 =$[CIR,O, P], 则非退化条件是 $A \neq B$, $O \neq P$.

例如在上面提到的例子中, 几何构造

```
[INTER, D, [PLINE, A, B, C], [PLINE, C, A, B]],
```

的等式型谓词是 [para, D, A, B, C], [para, D, C, A, B]; 非退化条件是 [para, B, C, A, B], 这一条件显然等价于 A, B, C 不共线.

在 MMP 中调用函数 ndg(gs), 返回几何命题 gs 的非退化条件. 如果 gs 是代数形式或谓词形式的几何命题, 则返回其非等式部分. 如果 gs 是构造形式或自然语言形式的几何命题, 则返回根据上面方法自动生成的非退化条件.

```
>ndg([[[POINT, A, B, C],
[INTER, D, [PLINE, A, B, C], [PLINE, C, A, B]],
[INTER, O, [LINE, A, C], [LINE, B, D]]], [[MIDPOINT, O, A, C]]]);
[[para, B, C, A, B], [para, A, C, B, D]]
```

这个几何命题的非退化条件是 BC 不平行于 AB, AC 不平行于 BD. 这个条件实际上就是 A, B, C 不共线.

4. 几何命题的代数形式. 最后介绍几何命题的代数形式. 几何定理机器证明的吴方法是一个基于代数运算的方法. 我们首先将一个需要证明的几何命题化为代数形式, 再用第四章中介绍的吴特征列方法来证明几何定理. 所以, 所有几何命题最后都将被转换为代数形式. 一个代数形式的几何命题如下给出：

```
Stat = [pvs, mvs, ps, ds, conc]
```

其中

- pvs 是命题中的非自由变量表. 变量的次序是由大到小.
- mvs 是命题中的自由变量表. 变量的次序是由大到小.
- ps 是命题中的等式型几何关系的多项式表.
- ds 是命题中的非等式型几何关系的多项式表.
- conc 是命题的结论.

例如：

```
[[], [x], [x^2-1], [x+1], [x-1]]
```

其意义如下：决定以下代数命题是否正确

$$(x^2 - 1 = 0 \wedge x + 1 \neq 0) \Rightarrow (x - 1 = 0)$$

几何命题的谓词形式可以直接转化为几何命题的代数形式. 我们只需要将相应的几何谓词翻译为代数公式即可. 表5.1是一些常用的几何谓词的代数表示, 其中的点 $P_i = (x_i, y_i)$.

表 5.1 几何谓词的代数表示

几 何 谓 词	代 数 表 示
[coll, P_1, P_2, P_3]	$(y_3 - y_1)(x_2 - x_1) - (x_3 - x_1)(y_2 - y_1) = 0$
[para, P_1, P_2, P_3, P_4]	$(y_4 - y_3)(x_2 - x_1) - (x_4 - x_3)(y_2 - y_1) = 0$
[perp, P_1, P_2, P_3, P_4]	$(y_4 - y_3)(y_2 - y_1) + (x_4 - x_3)(x_2 - x_1) = 0$
[cong, P_1, P_2, P_3, P_4]	$(y_4 - y_3)^2 + (x_4 - x_3)^2 - (y_2 - y_1)^2 - (x_2 - x_1)^2 = 0$
[dis, P_1, P_2, d]	$(x_1 - x_2)^2 + (y_1 - y_2)^2 - d^2 = 0$

调用 MMP 函数 geom(gs) 可以将任意形式的几何命题 gs 转换为代数形式.

```
>geom([[y_O, x_O, y_D, x_D], [v_C, u_C, v_B],
[A, [0,0], B, [0,v_B], C, [u_C,v_C], D, [x_D,y_D], O, [x_O,y_O]],
[[para, D, A, B, C], [para, D, C, A, B],  [coll, O, A, C],
[coll, O, B, D]],[[coll, A, B, C]], [[MIDPOINT, O, A, C]]]);
[[y_O, x_O, y_D, x_D], [v_C, u_C, v_B],
[u_C*y_D-v_C*x_D+v_B*x_D, -v_B*x_D+v_B*u_C,
u_C*y_O-v_C*x_O, x_D*y_O-y_D*x_O+v_B*x_O-v_B*x_D],
[-v_B*u_C], [2*x_O-u_C, 2*y_O-v_C]]
```

在以上例子中, 我们通过 MMP 的函数 geom 将例 5.1.1 的谓词形式转化成了代数形式.

§5.2 初等几何定理机器证明

1. 几何定理机器证明的吴方法. 对于初等几何中的定理, 适当选取坐标系, 可以将定理的条件和结论表示为多项式形式, 包括多项式等式与非等式：

$$\mathbb{H}=\{h_1(x_1,\cdots,x_n)=0,\cdots,h_r(x_1,\cdots,x_n)=0\}$$

$$\mathbb{D}=\{d_1(x_1,\cdots,x_n)\neq 0,\cdots,d_s(x_1,\cdots,x_n)\neq 0\}$$

可以表示为这种形式的几何命题称为 *等式型几何命题*. 定理的结论也可以表示为一个多项式方程 $C(x_1,\cdots,x_n)=0$. 其中非等式部分 $\mathbb{D}$ 称为几何命题的 *非退化条件*. 要证明一个几何定理, 就是要证明条件多项式组的所有零点都使得结论多项式为零：

$$\forall x_i[(h_1=0,\cdots,h_r=0,d_1\neq 0,\cdots,d_s\neq 0)\Rightarrow C=0]$$

等价的, 即证明 $\mathrm{Zero}(\mathbb{H}/\mathbb{D})\subset\mathrm{Zero}(C)$. 吴文俊提出了下面几何定理机器证明的基本原理.

几何定理机器证明原理 I. 给定一个定理的条件多项式组 $\mathbb{H}$ 和结论多项式 C. 设 $\mathcal{A}$ 为 $\mathbb{H}$ 的一个特征列, J 为 $\mathcal{A}$ 的初式乘积. 如果 $\mathrm{prem}(C,\mathcal{A})=0$, 则有

$$\mathrm{Zero}(\mathbb{H}/J)\subset\mathrm{Zero}(C) \tag{5.1}$$

也就是说定理在非退化条件 $J\neq 0$ 下成立. 如果 $\mathcal{A}$ 是一个不可约升列, 则 $\mathrm{prem}(C,\mathcal{A})=0$ 当且仅当 $\mathrm{Zero}(\mathbb{H}/J)\subset\mathrm{Zero}(C)$. 注意, 这里非退化条件 $J\neq 0$ 是自动生成的.

利用零点分解定理, 可以判定几何定理在某些分支上是否正确.

几何定理机器证明原理 II. 给定一个定理的条件多项式组 $\mathbb{H}$, $\mathbb{D}$ 和结论多项式 C. 运用吴特征列方法, 得到下面分解：

$$\mathrm{Zero}(\mathbb{H}/\mathbb{D}) \;=\; \cup_i\ \mathrm{Zero}(\mathcal{A}_i/\{I_i\}\cup\mathbb{D}) \tag{5.2}$$

其中 I_i 为升列 $\mathcal{A}_i$ 的初式乘积. 如果

$$\mathrm{prem}(C,\mathcal{A}_i)=0 \tag{5.3}$$

那么该几何定理在 $\mathrm{Zero}(\mathbb{H})$ 的 $\mathrm{Zero}(\mathcal{A}_i/I_i)$ 分支上正确. 如果对所有 i, (5.3) 成立, 则几何定理正确.

考虑例 5.1.1. 引入坐标 $A=[0,0],B=[0,v_B],C=[u_C,v_C],D=[x_D,y_D],O=[x_O,y_O]$. 可以得到几何条件 [para, D, A, B, C], [para, D, C, A, B], [coll, O, A, C], [coll, O, B, D] 与结论的代数形式.

```
>h1:=u_C*y_D-v_C*x_D+v_B*x_D;
>h2:=-v_B*x_D+v_B*u_C;
>h3:=u_C*y_O-v_C*x_O;
>h4:=x_D*y_O-y_D*x_O+v_B*x_O-v_B*x_D;
>c1:=2*x_O-u_C;
>c2:=2*y_O-v_C;
```

调用以下命令, 可以求得几何定理条件多项式 h_1, h_2, h_3, h_4 的特征列.

```
>vars:=[y_O, x_O, y_D, x_D, v_C, u_C, v_B, u_B, v_A, u_A];
>cs:=charset([h1, h2, h3, h4], vars);
cs:=[-v_B^2*u_C^2*v_C+2*y_O*v_B^2*u_C^2,-v_B^2*u_C^2+2*x_O*v_B^2*u_C,
-v_B*u_C*y_D+v_B*u_C*v_C-v_B^2*u_C,-v_B*x_D+v_B*u_C]
```

用结论多项式 $c1, c2$ 对升列 cs 做伪除, 所得余式为零.

```
>premas(c1, cs, vars);
0
>premas(c2, cs, vars);
0
```

根据机器证明原理 I, 例 5.1.1 在下面条件下正确:

$$v_B \neq 0 \wedge u_C \neq 0$$

以上代数条件的几何意义即 A, B, C 三点不共线.

现在用机器证明原理 II 证明例 5.1.1. 调用以下命令:

```
>wsolve([h1, h2, h3, h4], vars, [], "ritt");
```

得到六个升列:

```
cs1=[-2*y_O+v_C, -2*x_O+u_C, y_D-v_C+v_B, -x_D+u_C];
cs2=[u_C*y_O-v_C*x_O, u_C*y_D-v_C*x_D, v_B];
cs3=[u_C*y_O-v_C*x_O, y_D-v_C, -x_D+u_C, v_B];
cs4=[-x_D*y_O+y_D*x_O,v_C,u_C,v_B];
cs5=[y_D-v_B, x_D, v_C, u_C];
cs6=[x_O, x_D, u_C];
```

不难验证, 结论只在 cs1 所表示的分支上成立. 在其他分支上, A, B, C 三点共线. 此时, 几何定理失去意义, 结论不再正确. 如果调用以下命令:

```
>wsolve([h1, h2, h3, h4], vars, [v_B, u_C], "ritt");
```

则只得到 cs1 分支. 因此, 我们证明了下面几何定理的正确性.

例 5.2.1　A, B, C 是不共线的三点, $ABCD$ 是一平行四边形. 则其两对角线 AC,

BD 互相平分.

MMP 的命令 wprove(gs) 是根据机械化几何定理证明原理 I 与 II 编写的. 其中 gs 是一个几何命题. wprove 有两个主要步骤:

- 使用 geom(gs) 命令将几何命题转换为代数形式. 根据上一节的介绍, 如果 gs 是自然语言或者构造形式, 则 MMP 将自动生成非退化条件.
- 调用 wsolve 将几何命题的代数形式分解为三角列的形式, 并检查命题的结论在各个分支上是否正确.

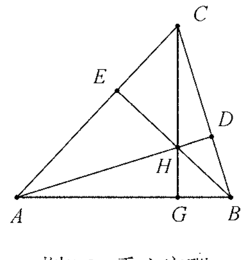
所以, wprove 证明几何命题正确是指其相应的谓词形式正确. 对于构造型命题与自然语言型命题, 系统自动添加了若干非退化条件. 例如, 第 §5.1 开始对于例 5.1.1 的证明实际上是证明例 5.2.1 的正确. 下面再举若干例子.

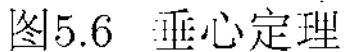
例 5.2.2 (垂心定理) 证明三角形的三条高相交于一点. 该点称为这一三角形的垂心 (图 5.6).

```
>wprove([[[POINT, A, B, C], [FOOT, D, A, B, C], [FOOT, E, B, A, C],
[INTER, H, [LINE, A, D], [LINE, B, E]]], [[PERP, A, B, C, H]]]);
The geometric Statement is valid!
```

例 5.2.3 (Ceva 定理) ABC 是一个三角形, D 是任意一点. 直线 DA, DB, DC 交三角形三边 BC, AC, AB 于 E, F, G. 证明 $BE/EC \cdot CF/FA \cdot AG/GB = 1$ (图 5.7).

```
>wprove([[[POINT, A, B, C, D],
[INTER, E, [LINE, A, D], [LINE, C, B]],
[INTER, F, [LINE, B, D], [LINE, C, A]],
[INTER, G, [LINE, C, D], [LINE, A, B]]],
[[NMINUS, [NPROD, [EQX, B, E], [EQX, C, F], [EQX, A, G]],
[NPROD, [EQX, E, C], [EQX, F, A], [EQX, G, B]]]]]);
The geometric Statement is valid!
```




图5.6 垂心定理

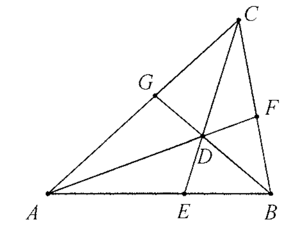


图5.7 Ceva定理

2. 非退化条件的充分性. 一个构造型几何命题称为 Hilbert 交点型命题, 如果在

构造几何元素时只使用直线, 以及中点, 比例点等可以构造唯一几何元素的几何构造. 同时命题的结论必须能转换为关于比例和面积的代数表达式的方程. 很显然, Hilbert 交点型命题是仿射几何中的构造型几何命题. 这类几何命题是由 Hilbert 在他的名著《几何基础》中定义的. 对于这类几何命题, Hilbert 给出了历史上第一个几何定理自动证明的算法. 我们这里考虑的构造型几何定理是 Hilbert 交点型命题的推广.

假定 HS 与 DS 是一个 Hilbert 交点型命题由 geomtopd 生成的等式部分与非等式部分, 则我们有

$$\mathrm{Zero}(HS/DS) = \mathrm{Zero}(\mathcal{A}/DS) \tag{5.4}$$

其中 A 是一个不可约升列. 这是因为, Hilbert 交点型命题所产生的代数方程关于主变量全部是线性的. 因此, 所得到的升列是不可约的. 关于上面等式的严格证明请见 [22].

如果一个几何命题具有性质 (5.4), 我们称之为 *不可约型几何命题*. 不可约型几何命题具有非常好的性质. 设相应几何命题的结论可以写为多项式 $C=0$. 假定已经得到 (5.4), 设 $R=\mathrm{prem}(C,\mathcal{A})$. 则有如下结论：

- 如果 $R=0$, 则几何命题正确.
- 如果 $R\neq 0$, 则几何命题不正确. 而且, 无论再添加什么样的非退化条件, 这一几何命题也不可能正确. 这是因为, 对于新的非退化条件 $d\neq 0$, 设 $DS_1 = DS\cup\{d\}$. 则

  $$\mathrm{Zero}(HS/DS_1) = \mathrm{Zero}(\mathcal{A}/DS_1)$$

  这里有两种情况：如果 $\mathrm{prem}(d,\mathcal{A})=0$, 则 $\mathrm{Zero}(\mathcal{A}/DS_1)=\emptyset$. 此时新添加的非退化条件与原几何命题矛盾. 如果 $\mathrm{prem}(d,\mathcal{A})\neq 0$, 则由以上分解以及条件 $\mathrm{prem}(C,\mathcal{A})\neq 0$, 新的几何命题仍然不正确. 也就是说, MMP 生成的非退化条件是 *充分的*.

除了 Hilbert 交点型命题以外, 还有一类几何命题是不可约的. 现定义如下：如果一个构造型几何命题仅使用下列几何构造, 则相应的几何命题是不可约的.

[on, o], 其中 o 是直线或圆.

[inter, o_1, o_2], 其中 o_1, o_2 是直线.

[inter, o_1, o_2], 其中 o_1 是直线, o_2 是圆, 并且 o_1 与 o_2 的一个交点已经知道. 例如, [inter, [line, A,B], [cir, O,A]] 是构造直线 AB 与圆 O 除 A 点之外的另一个交点. 所以由此导致的方程是线性的.

[inter, o_1, o_2], 其中 o_1, o_2 是圆, 并且 o_1 与 o_2 的一个交点已经知道. 例如, [inter, [cir, O_1,A], [cir, O_2,A]] 构造两个圆的另一个交点. 所以由此导致的方程是线

性的.

因为除了在圆上取任意点外, 其他所有的几何构造的代数方程关于主变量全部是线性的. 因此, 我们只需要证明由在圆上取任意点所产生的升列是不可约的. 具体讲, 我们需要证明下面的升列是不可约的.

$$
\begin{aligned}
A_1 &= (x_1 - a_1)^2 + (y_1 - b_1)^2 - r_1^2 \\
A_2 &= (x_2 - a_2)^2 + (y_2 - b_2)^2 - r_2^2 \\
&\cdots\cdots\cdots\cdots \\
A_n &= (x_n - a_n)^2 + (y_n - b_n)^2 - r_n^2
\end{aligned}
$$

其中 x_i, y_i 是变量, a_i, b_i, r_i 有可能是变量, 也有可能是数值. 我们规定 y_i 有较高的序, 则 $A_1, \cdots, A_n$ 显然是一个升列, 且可证明, 这是一个不可约升列 ([22]).

3. 最优非退化条件的生成. 几何定理的各种“非退化条件”是几何定理证明自动化的主要障碍. 使用演绎方法证明几何定理时, 非退化条件会导致所谓“分支”问题, 从而导致搜索空间爆炸. 实际上, 非退化条件的存在也是几何传统证明方法的一个主要问题. 下面引用吴文俊的一段话加以说明 ([87]).

> 在 Euclid 几何中, 公理与定理的叙述往往隐含了一种没有明白说出的假设 — 所考虑的图形必须处在某种正常的、一般性的、而非异常的、带有特殊性的退化情况. 例如, 在说到两直线平行时, 就隐含着它们是不同的两条直线而不是退化为两条重合的直线. 同样在说到两条直线相交时, 也隐含着它们并没有退化为两条平行的或重合的直线. 又如在说到三角形时, 总是隐含了这是一个正常的真正三角形, 它的三个顶点互不相同且不退化为顶点在同一直线上的一个“扁”三角形, 如此等等. 虽然我们可在定义或定理的叙述中加上种种非退化的限制, 但那样叙述显得极为冗沓. 究竟给出什么样的非退化的限制才算合适并不清楚. 退化一词也没有确切的定义, 所以等腰三角形或直角三角形算不算是一个退化的三角形, 就很难确定了.
>
> 尤其严重的是, 定理的证明往往只适用于某种正常的、一般的、而非异常的或退化的情形. 对于退化的情形, 往往对证明作必要的修改才能适用. 或者需要改变全部证明. 有时对于退化情形定理本身甚至完全失去意义以至根本不成立.

在 MMP 中, 用户可以使用三种方法处理非退化条件：对于用自然语言或者构造性语言描述的几何命题, 其非退化条件是由 MMP 自动生成的. 而且, 我们已经证明对于不可约型几何命题, 这样生成的非退化条件是充分的. 在用谓词形式或代数形式描述的几何命题中, 非退化条件是由用户自己指定的. 下面我们将介绍怎样用 MMP 生成某种最优非退化条件.

对一个几何定理, 我们将其中涉及的坐标分为两类：参变量 $x_1,\cdots,x_m$ 与主变量 $x_{m+1},\cdots,x_n$. 其中参变量的值在几何命题中是可以任意选取的, 而主变量的值则由参变量的值确定. 一个几何定理的条件多项式组是 $\mathbb{H}$ 与 $\mathbb{D}$, 结论多项式是 C. 那么拟代数簇 Zero($\mathbb{H}/\mathbb{D}$) 在假定参变量 $x_1,\cdots,x_m$ 上的投影

$$\mathrm{NDG} = \mathrm{Proj}_{x_1,\cdots,x_m} Zero(\mathbb{H}/\mathbb{D}\cup\{C\})$$

就表示当定理不成立时, 参变量 $x_1,\cdots,x_m$ 满足的充分必要条件. 而投影结果的补集可被看作是几何定理成立时充分必要的非退化条件.

更进一步, 因为参变量的值在几何命题中是可以任意选取的, 如果 NDG 的维数大于零, 则说明几何命题不成立的情况所对应的参数值是总的参数空间的一个低维子集. 也就是说：几何定理成立的参数值是整个参数空间去掉一些低维子集. 此时, 我们说几何命题是 *一般正确的*.

MMP 的函数 tprove(gs) 实现了以上投影法证明几何定理的原理. 其中 gs 是一个谓词或代数形式的几何命题. 这一命令将自动生成定理不成立时关于参变量的充要条件.

例 5.2.4　*四边形 $ABCD$ 是一个平行四边形, 如图 5.8 所示. P 是该平面上任意一点. 那么三角形 PAB, PCD 和 ABC 的面积关系为 $S_{ABC}-S_{PCD}-S_{PAB}=0$.*

引入坐标 A=[0, 0], B=[x1, 0], C=[x2, x3], D=[x4, x3], P=[x5, x6]. 可以得到几何条件 [para, A, D, B, C] , [area, s1, P, A, B] [area, s2, P, C, D], [area, s3, A, B, C] 和结论的代数形式：

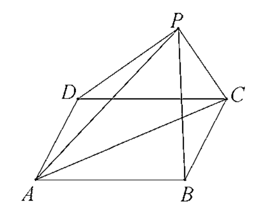
```
>h1:=x3*x4-x3*(-x1+x2);
>h2:=2*s1-x6*x1;
>h3:=2*s2-(x2-x5)*x3-(-x3+x6)*x4-x5*x3+x6*x2;
>h4:=2*s3-x1*x3;
>c:=s3-s2-s1;
```

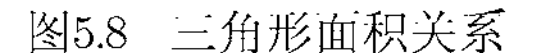
调用以下命令可以求得几何定理不成立时关于 x1, x2, x3, x4 的充要条件：

```
>tprove([[x4,x3,x2,x1], [s3,s2,s1,x6,x5], [h1,h2,h3,h4], [], [c]]);
The sufficient and necessary conditions for the theorem to be FALSE:
[[[x3], [x4+x1-x2]]]
```

该结果 x3=0 且 x4+x1-x2≠0 表示当点 C 和 A,B 共线, 且线段长度 $|AB|$ 不等于 $|CD|$ 时, 关系式 s3-s2-s1=0 不成立. 图 5.8 表示一般情形定理成立, 图 5.9 表示 A, B, C 共线的退化情形, 但此时 $|AB|=|CD|$, 故定理仍成立.

因为 Zero($x3/x4+x1-x2$) 的维数为一, 这一几何命题是一般正确的.




图5.8 三角形面积关系

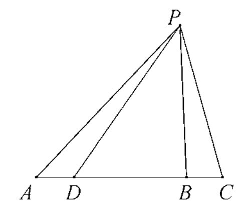


图5.9 三角形ABC退化但定理成立

§5.3 初等几何定理自动发现

1. 公式自动发现的吴方法. 除了自动证明几何定理外, 吴特征列方法还可以用来自动发现未知的几何关系. 基本原理如下. 假设一个几何问题的条件可以用下列多项式等式与非等式表示：

$$\mathbb{H}=\{h_1(u_1,\cdots,u_d,x_1,\cdots,x_n)=0,\cdots,h_r(u_1,\cdots,u_d,x_1,\cdots,x_n)=0\}$$

$$\mathbb{D}=\{d_1(u_1,\cdots,u_d,x_1,\cdots,x_n)\neq 0,\cdots,d_s(u_1,\cdots,u_d,x_1,\cdots,x_n)\neq 0\}$$

这里, 我们将变量分为两类：$\mathbb{U}=\{u_1,\cdots,u_d\}$ 在几何问题中可以取任意值, 称为 *自由变量*, $\mathbb{X}=\{x_1,\cdots,x_n\}$ 的值则依赖于 $\mathbb{U}$ 的值. 严格讲, 我们要求除了 $\mathbb{U}$ 的一些特殊值之外, 每一个 x_i 可以由 $\mathbb{U}$ 的值具体确定. 将 h_i, d_i 看成 $\mathbb{Q}(\mathbb{U})[x_1,\cdots,x_n]$ 中的多项式, 在变量顺序 $x_1<\cdots<x_n$ 下应用吴特征列方法, 我们有

$$\mathrm{Zero}(\mathbb{H}/\mathbb{D})=\cup_{i=1}^{s}\mathrm{Zero}(\mathrm{sat}(\mathcal{A}_i)/\mathbb{D})$$

由于 $\mathbb{U}$ 是自由变量, 每一个 $\mathcal{A}_i$ 中的第一个多项式应该具有形式

$$R_i(u_1,\cdots,u_d,x_1)=0$$

这一方程给出了自由变量 $\mathbb{U}$ 与变量 x_1 之间的关系. 我们可以进一步证明

- 这样推导的几何公式 $\{R_1(\mathbb{U},x_1),\cdots,R_s(\mathbb{U},x_1)\}$ 存在而且是唯一的.
- 存在一个只包含 $\mathbb{U}$ 的多项式 D, 使得在 $D\neq 0$ 的条件下, $R_i=0$ 可以由 $\mathbb{H}$ 与 $\mathbb{D}$ 推得. 具体讲, 我们有

$$\forall x_i,u_j[(\mathbb{H}\wedge\mathbb{D}\wedge D\neq 0)\rightarrow(R_1=0\vee\cdots\vee R_s=0)]$$

MMP 的函数 wderive(gs) 实现了以上公式自动发现的原理. 其中 gs 是一个谓词形式或代数形式的几何命题. 这一命令将用吴方法自动推导最后一个主变量与所有参变量之间的关系.

2. 公式自动发现举例. 下面介绍几类公式推导问题.

几何公式的推导. 几何中常见的关于面积、体积的公式可以用这一方法自动求解. 举例说明如下.

例 5.3.1 (Heron- 秦公式) *试决定三角形的面积与其三条边之间的关系.*

```
>wderive([[x2,x1,k], [c,b,a], [b1, [0,0], c1, [a,0], a1, [x1,x2]],
[[dis, a1,c1,b], [dis, a1,b1,c], [area,k,a1,b1,c1]], [], []]);
Formula 1 :16*k^2+a^4-2*b^2*a^2-2*c^2*a^2+b^4-2*c^2*b^2+c^4
1 formulas found.
```

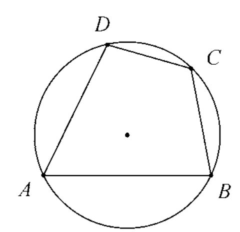
这里, 我们假定三角形的三条边长 c, b, a 是自由变量, 面积 k 是次序最低的一个主变量. 上面自动求得的公式就是熟知的 Heron- 秦公式：

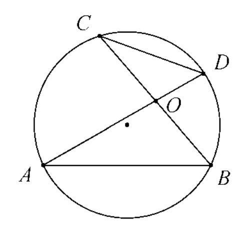
$$k = \sqrt{(s(s-a)(s-b)(s-c))}$$

其中 $s = (a+b+c)/2$.

例 5.3.2 (Brahmagupta 公式) *$ABCD$ 是一个圆内接四边形. 求它带符号的面积关于其四边的表达式* (图 5.10).

```
>wderive([[x4, x3, x2, x1, k], [u1, u2, u3, u4],
[a, [0, 0], b, [u1, 0], c, [x1, x2], d, [x3, x4]],
[[cir, a,b,c,d], [dis, b,c,u2], [dis, c,d,u3], [dis, d,a,u4],
[area, k, a, b, c, d]], [], []]);
Formula 1:
16*k^2+u1^4-2*u2^2*u1^2-2*u3^2*u1^2-2*u4^2*u1^2+8*u4*u3*u2*u1
+u2^4-2*u3^2*u2^2-2*u4^2*u2^2+u3^4-2*u4^2*u3^2+u4^4
Formula 2:
16*k^2+u1^4-2*u2^2*u1^2-2*u3^2*u1^2-2*u4^2*u1^2-8*u4*u3*u2*u1
+u2^4-2*u3^2*u2^2-2*u4^2*u2^2+u3^4-2*u4^2*u3^2+u4^4
2 formulas found.
```





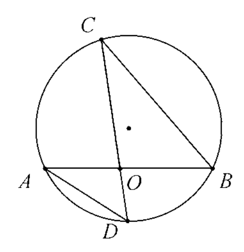


图 5.10　Brahmagupta 公式的三种情形

这里给出两个公式：

$$k = \sqrt{(s-u_1)(s-u_2)(s-u_3)(s-u_4)}$$
$$k = \sqrt{s(s-u_1-u_3)(s-u_1-u_2)(s-u_1-u_4)}$$

其中 $s=(u_1+u_2+u_3+u_4)/2$. 第一个公式即熟知的 Brahmagupta 公式. 第二个公式, 实际上是第一个的某种“反射”. 如果一个或三个 u_i 取负值, 则第二个公式给出四边形的面积. 在其他情形, 第一个公式给出四边形的面积. 之所以产生这样的情形是因为吴方法没有区别变量 u_i 的正负, 所以需要给出所有可能的结论. 吴方法的这一特性其实是它的优点. 例如, 正是因为没有考虑次序关系, 我们推导的两个公式对于图 5.10 中的三种情形都正确. 在第二个图中, k 是 $\triangle ABO$ 与 $\triangle CDO$ 的有向面积的和. 在第三个图中, k 是 $\triangle BCO$ 与 $\triangle DAO$ 的有向面积的和.

几何定理的发现. 对于几何图形中的一些几何量, 我们可以用 MMP 推导它们之间的关系, 从而自动发现这些几何定理.

下面是 Ceva 定理 (例 5.2.3) 的自动推导.

```
>wderive([[x10, x9, x8, x7, x6, x5, x2, r3], [x1, x3, x4, r1, r2],
[d, [0,0], b, [x1,0], c, [x2,0], a, [x3,x4], e, [x5,x6], f, [x7,x8],
g, [x9, x10]], [[coll, d, g, a], [coll, b, f, a], [coll, c, e, a],
[coll, c, g, f], [coll, b, g, e], [xrat, r1, b, d, d, c],
[xrat, r2, c, e, e, a], [xrat, r3, a, f, f, b]], [], []]);
Formula 1 :r2*r1*r3-1
1 formulas found.
```

例 5.3.3 (Menelaus 定理) *ABC 是一个三角形. 一条直线交三角形三边 BC, AC, AB 于 E,F,G. 证明 BE/EC, CF/FA, AG/GB 之间的关系* (图 5.11).

使用 MMP 的 wderive 函数, 可以自动推导出三个比例之间的关系.

```
>wderive([[x8, x7, x6, x5, x4, x3, x2, x1, r3], [r2, r1],
[a, [0,0], b, [x1,0], c, [x2,x3], g, [x4,0], e, [x5,x6], f, [x7,x8]],
[[coll, b, c, e], [coll, c, a, f], [coll, e, f, g],
[xrat, r1,b,e,e,c], [xrat, r2,c,f,f,a], [xrat, r3,a,g,g,b]],
[[coll, a, b, c]], []]);
Formula 1 :r2*r1*r3+1
Formula 2 :r2*r3-1
Formula 3 :r1*r3-1
3 formulas found.
```

这样, 我们就自动发现了 Melenaus 定理：$BE/EC \cdot CF/FA \cdot AG/GB = -1$.

例 5.3.4　设 I 是三角形 ABC 的内心, I_1 是三角形 ABC 关于角 A 的外心, D 是 BC 与 II_1 的交点. 求交比 (AD, II_1)(图 5.12).

```
>wderive([[x5, x4, x3, x2, x1, r], [u1, u2, u3],
[b,[0,0], c,[u1,0], i,[u2,u3], a,[x1,x2], d,[x3,0], i1,[x4,x5]],
[[eqtang, c, b, i, i, b, a], [eqtang, b, c, i, i, c, a],
[coll, a, i, d], [coll, i1, a, i], [perp, b, i, b, i1],
[crat, r, a, d, i, i1]],  [], []]);
Formula 1 : r+1
1 formulas found.
```

由所给结果不难发现, 交比 $(AD, II_1) = -1$, 即 A, D, I, I_1 四点成调和点列.

例 5.3.5　设 I 是三角形 ABC 的内心, E 是直线 AI 与三角形 ABC 外接圆的交点, D 是 AI 与 BC 的交点. 求 AB, AC, AD, AE 之间的关系 (图 5.13).

```
>wderive([[x5, x4, x3, x2, x1, r4], [r3, r2, r1],
[a,[0,0], c,[x1,x2], f,[x1,x3], b,[x4,x5], d,[r1,0], e,[r2,0]],
[[nsum, x3, x2], [coll, a, f, b], [coll, d, b, c],
[CYCLIC, a, b, c, e], [dis, a, b, r3], [dis, a, c, r4]],
[[coll, a, b, c]], []]);
Formula 1 :-r3*r4+r1*r2
Formula 2 :r3*r4+r1*r2
2 formulas found.
```

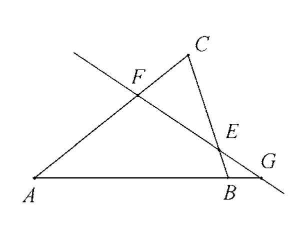


图5.11　Menelaus定理

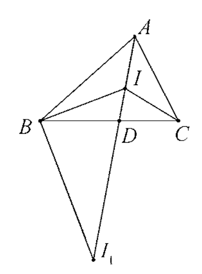


图5.12　三角形内心与外心

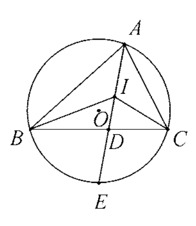
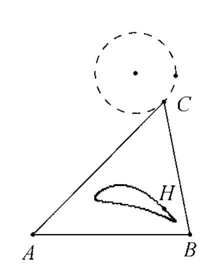


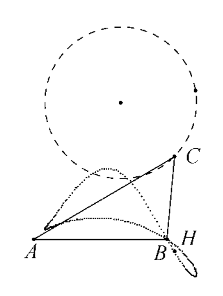
图5.13　三角形内心与外接圆

因此, 我们发现两个关系：$AD*AE = \pm AB*AC$. 由于边长为正数, 只有 $AD*AE = AB*AC$ 有意义.

几何轨迹的自动发现. 几何轨迹实际上是一个点的坐标与其他若干几何量之间的关系. 因此, 也可以用 MMP 自动发现. 我们在例1.5.3中给出的 Peaucellier 连杆就是几何轨迹自动发现的例子.

例 5.3.6　三角形两点固定, 第三点在一个圆上运动. 求其垂心的轨迹 (图 5.14).





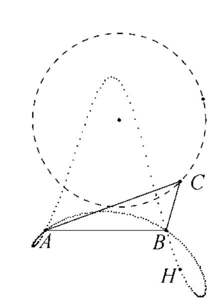


图 5.14 三角形垂心的轨迹

使用 wderive 很快就可以得出这一轨迹是一条四次曲线. 使用 "几何专家" 软件还可以进一步动态显示这条轨迹对于圆的半径的依赖关系 (图 5.14).

```
>wderive([[y1, x1, y], [x, a, u, v, r],
[A, [0, 0], B, [a, 0], C, [x1, y1], H, [x, y], O, [u, v]],
[[perp, A, H, B, C], [perp, B, H, A, C], [dis, O, C, r]], [], []]);
Formula 1 :x^2*a^2-2*x^3*a-2*v*y*x*a-y^2*r^2+x^4+y^2*x^2+2*v*y*x^2
          -2*u*y^2*x+u^2*y^2+v^2*y^2
1 formulas found.
```

3. 完整几何公式的自动发现. 在本节开始所叙述的方法, 我们只给出了 "主要的几何关系", 对于某些特殊情形没有考虑. 更加明确地讲, 我们只考虑参数 $\mathbb{U}$ 互相独立情形 x_1 与 $\mathbb{U}$ 的关系. 为了得到完整的几何关系, 我们需要使用第四章引入的投影定理.

给定一个几何命题的条件 $\mathbb{H}$ 和 $\mathbb{D}$, 这里 $\mathbb{H}$ 和 $\mathbb{D}$ 分别是如前所述的多项式等式与非等式. 那么拟代数簇 $\mathrm{Zero}(\mathbb{H}/\mathbb{D})$ 在自由变量 $\mathbb{U}$ 上的投影

$$\mathrm{Proj}_{x_1,\cdots,x_n}\mathrm{Zero}(\mathbb{H}/\mathbb{D})$$

就表示为在条件 $\mathbb{H}$ 和 $\mathbb{D}$ 下, 自由变量 $\mathbb{U}$ 之间的关系, 其中可能既有等式又有不等式.

MMP 的函数 tderive(gs) 实现了以上投影法公式自动发现的原理. 其中 gs 是一个代数形式的几何命题. 这一命令将自动推导出所有参变量之间的关系.

例 5.3.7 用 tderive 推导出例 5.2.4中的三角形面积关系式.

```
>h1:=x3*x4-x3*(-x1+x2);
>h2:=2*s1-x6*x1;
>h3:=2*s2-(x2-x5)*x3-(-x3+x6)*x4-x5*x3+x6*x2;
>h4:=2*s3-x1*x3;
>tderive([[s3,s2,s1], [x6,x5,x4,x3,x2,x1], [h1,h2,h3,h4],[],[]]);
```

```
[[[s3], [s1]], [[s1, s3], [1]], [[s1+s2-s3], [1]]]
```

该结果表示 s3=0 且 s1≠0, 或者 s1=0 且 s3=0, 或者 s1+s2−s3=0. 前两个分支合在一起就是 s3=0. 综合起来的几何意义为三角形面积关系满足等式 s1+s2−s3=0; 或者三角形 ABC 面积为零, 此时另两个三角形的面积可为任意值.

§5.4　微分几何定理机器证明与发现

本节介绍微分几何定理机器证明与发现.

1. 微分几何定理机器证明的吴方法. MMP 中用来证明微分几何定理的函数是 wdprove(gs), 其中 gs 是一个代数形式或谓词形式的微分几何命题. 这一命令将用吴方法 (基于零点分解定理的完全方法) 证明几何命题的代数形式是否正确. 微分几何命题的证明原理与初等几何相似, 这里不再重复. 我们在例 5.1.2 中使用这一方法由 Kepler 行星运动定理自动证明了 Newton 万有引力定理.

下面介绍两个技巧. 第一, 如果某些变量是常数, 则我们可以不将它们包含在 depend 命令中, 这样 MMP 会自动将这些变量当作常数. 第二, 如果 x 是 s 的函数, x[2] 与 diff(x, s, 2) 具有相同的意义. 利用这些技巧, 我们得到例 1.5.2 的一个等价描述：

```
>depend([a, r, y, x], [t]);
>wdprove( [[a, r, y, x, p, e], [], [],
[r^2-x^2-y^2, a^2-x[2]^2-y[2]^2, x*y[2]-x[2]*y, r-p-e*x],
[p], [diff(a*r^2, t)]]);
```

为了由 Newton 万有引力定理 N1 证明 Kepler 行星运动定理 K1, 我们需要引入 wronskian 函数. 这一函数基于下面定理.

定理 5.4.1　*假设 $f_1,\cdots,f_n$ 是变量 t 的函数. 用 $f_i^{(k)}$ 表示 f_i 关于 t 的 k 阶导数. 则存在非零常数 $c_i, i=1,\cdots,n$, 使得 $\sum_{i=1}^n c_i f_i = 0$ 当且仅当*

$$W=\begin{vmatrix} f_1 & f_2 & \cdots & f_n \\ f_1^{(1)} & f_2^{(1)} & \cdots & f_n^{(1)} \\ \vdots & \vdots & & \vdots \\ f_1^{(n-1)} & f_2^{(n-1)} & \cdots & f_n^{(n-1)} \end{vmatrix}=0$$

*W 称为函数 $f_1,\cdots,f_n$ 的*wronskian.

MMP 中, 用 wronskian(ps, v) 计算 wronskian. 其中 ps 是一个微分多项式列表, v 是微分变量. 例如, 点 (x, y, z) 在一个固定平面上当且仅当存在常数 a,b,c,d , 使得 $ax+by+cz+d=0$. 由定理 5.4.1, 这一条件可以计算如下：

```
>depend([x, y, z], [t]);
>wronskian([1, x, y, z], t);
z[3]*y[2]*x[1]-y[3]*z[2]*x[1]-z[3]*y[1]*x[2]+y[3]*z[1]*x[2]
+x[3]*z[2]*y[1]-x[3]*z[1]*y[2]
```

现在我们可以由 Newton 定理 N1 证明 Kepler 行星运动定理 K1.

```
>depend([a, r, x, y], [t]);
>wdprove([[a, r, y, x], [], [],
[r^2-x^2-y^2,a^2-x[2]^2-y[2]^2,x*y[2]-x[2]*y,diff(a*r^2, t)],
[],[wronskian([1, x, y, r], t)]]);
The geometric Statement is valid!
19 components checked.
```

基于 wronskian 函数, 我们给出一套关于微分曲线的几何语言. 具体讲即引入了若干关于曲线的微分几何谓词. 举例如下.

假设 $C = [x, y, z]$ 是一个三元组, 代表一条空间曲线的坐标. 在 MMP 中可以使用的谓词见附录 A. 一些常见谓词如下:

[FIX_PLANE, C]	C 在一个固定平面上
[FIX_SPHERE, C]	C 在一个固定球面上
[FIX_HELIX, C]	C 在一个固定螺旋面上
[XY_CIRCLE, C]	C 在 XY 平面上的投影在一个圆上

2. 空间曲线定理的机器证明. MMP 的函数 curve() 可以产生曲线 $C(t) = (x(t), y(t), z(t))$ 的下列基本不变量及其定义.

$$
\begin{aligned}
&z'^2 + y'^2 + x'^2 - 1 = 0\\
&k^2 - z''^2 - y''^2 - x''^2 = 0, kr - 1 = 0\\
&kn_1 - x'' = 0, kn_2 - y'' = 0, kn_3 - z'' = 0\\
&kb_1 - y'z'' + y''z' = 0, kb_2 + x'z'' - x''z' = 0, kb_3 - x'y'' + x''y' = 0\\
&o_1 - rn_1 - x = 0, o_2 - rn_2 - y = 0, o_3 - rn_3 - z = 0\\
&t + n_3b_3' + n_2b_2' + n_1b_1' = 0
\end{aligned} \tag{5.5}
$$

其中 k 是曲线的曲率, t 是曲线的绕率, (n_1, n_2, n_3) 是曲线的法向量. 我们将假定 $k \neq 0$, 也就是说假定所考虑的曲线不是直线.

在调用函数 curve() 后, 我们可以用 wprove_curve(gs) 证明几何定理, 其中 gs 是一个谓词形式的微分几何命题. 这一命令用零点分解定理将几何命题的假设微分方程分解为三角形式, 并验证几何命题在所有分支上是否正确. 在使用这一命令前必须运行 curve(). 假设 gs = [pvs, mvs, pts, ps, ds, conc], 这一命令实际证明的是如

下命题：
gs1 = [pvs, mvs1, pts1, ps1, ds, conc], 其中

$$\begin{aligned}\text{mvs1} &= \text{mvs} \cup [t, o_3, o_2, o_1, b_3, b_2, b_1, n_3, n_2, n_1, r, k, z, y, x, s]\\ \text{pts1} &= \text{pts} \cup [C, [x, y, z], T, [x', y', z'], N, [n_1, n_2, n_3], B, [b_1, b_2, b_3], O, [o_1, o_2, o_3]]\\ \text{ps1} &= \text{ps} \cup (5.5)\end{aligned}$$

例 5.4.2 一条空间曲线是圆当且仅当 $k' = t = 0$.

```
>curve();
>wprove_curve([[], [], [], [z, [XY_CIRCLE, C]], [], [t, k[1]]]);
The geometric Statement is valid!
1 components checked.
>wprove_curve([[],[],[],[t,k[1]],[],[[FIX_PLANE,C],[FIX_SPHERE,C]]]);
The geometric Statement is valid!
2 components checked.
```

第一次调用 wprove_curve(gs) 证明了圆满足给定条件. 第二次调用则证明满足条件的曲线是圆.

如果需要证明关于多条空间曲线的几何定理, 可以在 curve() 定义的曲线基础上用谓词 curve 添加新的空间曲线, 同时必须把新增加的微分变元加入依赖关系.

例 5.4.3 如果两条曲线关于一个平面对称, 那么在相应点曲率相等, 挠率互为相反.

```
>curve();
>adepend([x1, y1, z1, k1, k2, t2]);
>wprove_curve([[x1, y1, z1, k1, k2, t2], [], [v1, [x1, y1, z1]],
[x1-x,y1-y,z1+z, [curve,v1,k1,k2,t2]], [k1,k2], [t2+t,k^2-k2^2]]);
The geometric Statement is valid!
2 components checked.
```

例 5.4.4 对于非直线的曲线, 下列叙述等价：

(a) 挠率与曲率的比值为常数.

(b) 曲线关于一条固定直线成一个固定角度 C.

(c) 曲线的主法线平行于一固定平面 N.

(d) 曲线的次法线相对于固定直线成固定角度 B.

满足这些条件的曲线称为螺旋线. 由定理 5.4.1, $(a), (b), (c)$ 和 (d) 分别等价于：

```
>h1:=wronskian([k, t], s);
>h2:=wronskian([1, diff(x, s), diff(y, s), diff(z, s)], s);
```

```
>h3:=wronskian([diff(x, s, 2), diff(y, s, 2), diff(z, s, 2)], s);
>h4:=wronskian([1, b1, b2, b3], s);
```

很明显, h2=h3. 下面三个命令分别证明 $(a) \Rightarrow (b), (b) \Rightarrow (d), (d) \Rightarrow (a)$.

```
>curve();
>wprove_curve([[], [], [], [h1], [], [h2]]);
>wprove_curve([[], [], [], [h2], [], [h4]]);
>wprove_curve([[], [], [], [h4], [], [h1]]);
```

3. 微分几何定理的自动发现. 在微分几何中自动发现定理与初等几何类似. 假设一个几何问题的条件可以用下列微分方程等式与非等式表示：

$$\mathbb{H} = \{h_1(u_1, \cdots, u_d, x_1, \cdots, x_n) = 0, \cdots, h_r(u_1, \cdots, u_d, x_1, \cdots, x_n) = 0\}$$

$$\mathbb{D} = \{d_1(u_1, \cdots, u_d, x_1, \cdots, x_n) \neq 0, \cdots, d_s(u_1, \cdots, u_d, x_1, \cdots, x_n) \neq 0\}$$

我们同样将变量分为两类：$\mathbb{U} = \{u_1, \cdots, u_d\}$ 与 $\mathbb{X} = \{x_1, \cdots, x_n\}$. 注意此时我们并没有要求 $\mathbb{U}$ 是独立变量. 在变量顺序 $u_1 < \cdots < u_d < x_1 < \cdots < x_n$ 下应用零点分解定理, 我们有

$$\text{Zero}(\mathbb{H}/\mathbb{D}) = \cup_{i=1}^{s} \text{Zero}(\text{sat}(\mathcal{A}_i)/\mathbb{D})$$

如果 $\mathbb{U}$ 是自由变量, 每一个 $\mathcal{A}_i$ 中的第一个多项式应该具有形式

$$R_i(u_1, \cdots, u_d, x_1) = 0$$

这一方程给出了自由变量 $\mathbb{U}$ 与变量 x_1 之间的关系.

例 5.4.5 计算一条直线的曲率.

记曲线为 $C = (x, y, z)$, 则问题可以表示为 $(\mathbb{H}, \mathbb{D})$, 其中 $\mathbb{D} = \emptyset$, $\mathbb{H} = \{h_1, \cdots, h_5\}$.

```
>depend([z, y, x, k], [s]);
>h1:=z[1]^2+y[1]^2+x[1]^2-1;
>h2:=z[2]^2+y[2]^2+x[2]^2-k^2;
>h3:=wronskian([1, y, z], s);
>h4:=wronskian([1, x, y], s);
>h5:=wronskian([1, x, z], s);
>dwsolve([h1, h2, h3, h4, h5], [z, y, x, k], []);
[z[1]^2+y[1]^2-1, y[2], x[1], k], [z[1], y[1]+1, x[1], k],
[z[1], y[1]-1, x[1], k], [z[1]^2+y[1]^2+x[1]^2-1, y[2], x[2], k],
[z[1], y[1]^2+x[1]^2-1, x[2], k], [z[1], y[1], x[1]+1, k],
```

```
[z[1], y[1], x[1]-1, k]
```

这一问题没有参数变量. 利用零点分解算法, 取变量序为 $k < x < y < z$, 我们有 $\mathrm{Zero}(\mathbb{H}) = \cup_{1\leqslant i\leqslant 6}\mathrm{Zero}(\mathrm{sat}(\mathcal{A}_i))$ ，其中

$$
\begin{array}{lllll}
\mathcal{A}_1 = & k & x' & y'' & z'^2+y'^2-1 \\
\mathcal{A}_2 = & k & x' & y'+1 & z' \\
\mathcal{A}_3 = & k & x' & y'-1 & z' \\
\mathcal{A}_4 = & k & x'' & y'' & z'^2+y'^2+x'^2-1 \\
\mathcal{A}_5 = & k & x'' & y'^2+x'^2-1 & z' \\
\mathcal{A}_6 = & k & x'-1 & y' & z'
\end{array}
$$

关系集合为 $\{k\}$. 也就是说, 对于直线 $k = 0$.

例 5.4.6　我们现在自动发现例 5.4.4 的一些结果. 一非直线的曲线关于一条固定直线成一个固定角度. 求其挠率与曲率之间的关系.

```
>depend([t, n3, n2, n1, b3, b2, b1, k, z, y, x], [s]);
>h1:=z[1]^2 + y[1]^2 + x[1]^2 - 1;
>h2:=k^2 - z[2]^2 - y[2]^2 - x[2]^2;
>h3:=k*n1 - x[2];
>h4:=k*n2 - y[2];
>h5:=k*n3 - z[2];
>h6:=k*b1 - y[1]*z[2] + y[2]*z[2];
>h7:=k*b2 + x[1]*z[2] - x[2]*z[1];
>h8:=k*b3 - x[1]*y[2] + x[2]*y[1];
>h9:=t + n3*b3[1] + n2*b2[1] + n1*b1[1];
>h0:=wronskian([1, diff(x, s), diff(y, s), diff(z, s)], s);
>vs:=[n3, n2, n1, b3, b2, b1, z, y, x, t, k];
>dwsolve([h0, h1, h2, h3, h4, h5, h6, h7, h8, h9], vs, [k]);
```

例 5.4.7 (Bertrand 曲线)　具有共同的主法向量的空间曲线称为相关的Bertrand 曲线对. 求 Bertrand 曲线的曲率和挠率之间的关系.

给定两条一一对应的空间曲线 C_1 和 C_2, 将移动三角架 $(C_1, e_{11}, e_{12}, e_{13})$ 和 $(C_2, e_{21}, e_{22}, e_{23})$ 赋予 C_1 和 C_2. 记 C_1 和 C_2 的弧长、曲率和挠率分别为 s_1, k_1, t_1 和 s_2, k_2, t_2. 则上述给出的参量可以看成 s_1 的函数. 令 $r = \frac{ds_2}{ds_1}$. 我们有 $C_2 = C_1 + a_2 e_{12}$, 且

$$e_{21} = u_{11}e_{11} + u_{13}e_{13}$$

$$e_{22} = e_{12}$$

$$e_{23} = -u_{13}e_{11} + u_{11}e_{13}$$

其中 $h_1 = u_{13}^2 + u_{11}^2 - 1 = 0$. 利用 Frenet 公式可将问题表示为 $(\mathbb{H}, \mathbb{D})$.

$\mathbb{H} = \{h_1, \cdots, h_{10}\}$, 其中

$h_2 = ru_{13} - t_1a_2$

$h_3 = a_2'$

$h_4 = ru_{11} + k_1a_2 - 1$

$h_5 = u_{13}'$

$h_6 = rk_2 + t_1u_{13} - k_1u_{11}$

$h_7 = ru_{11}t_2 - ru_{13}k_2 - t_1$

$h_8 = ru_{13}t_2 + ru_{11}k_2 - k_1$

$h_9 = u_{11}'$

$h_{10} = rt_2 - k_1u_{13} - t_1u_{11}$

$\mathbb{D} = \{k_1, t_1, k_2, t_2, a_2\}$

用吴特征列方法, 我们可以得到 k_1 和 t_1 之间的一组关系集 $\{k_1't_1'' - k_1''t_1'\}$.

```
>depend([u13, u11, r, a2, k2, k1, t2, t1], [s]);
>h1 := u13^2 + u11^2 - 1;
>h2 := r*u13 - t1*a2;
>h3 := r*u11 + k1*a2 - 1;
>h4 := r*k2 + t1*u13 - k1*u11;
>h5 := r*u11*t2 - r*u13*k2 - t1;
>h6 := r*u13*t2 + r*u11*k2 - k1;
>h7 := r*t2 - k1*u13 - t1*u11;
>dwsolve([h1, h2, h3, h4, h5, h6, h7, a2[1], u13[1], u11[1]],
[t1, k1, t2, k2, a2, r, u13, u11], [k1, t1, k2, t2, a2]);
```

例 5.4.8 由 Kepler 定律自动推导 Newton 引力定律.

在变量顺序 $y > x > a > r$ 下使用零点分解定理:

```
>vs:=[y, x, a, r];
>depend(vs, [t]);
>ps:=[x*y[2]-x[2]*y, r^2-x^2-y^2, a^2-x[2]^2-y[2]^2,
wronskian([1, r, x], t)];
>dwsolve(ps, vs, [a, x[1]*y[2]-x[2]*y[1]], "weak");
```

我们有 $\mathrm{Zero}(\mathbb{H}/\mathbb{D}) = \cup_{1\leqslant i\leqslant 4}\mathrm{Zero}(\mathrm{sat}(\mathcal{A}_i)/\mathbb{D})$, 其中
$\mathcal{A}_1 = \{y^2 + x^2 - r^2, a * x^2 - r^2 * a - x'^2 * r, a', r'\}$
$\mathcal{A}_2 = \{y^2 + x^2 - r^2, a * x^2 - r^2 * a + x'^2 * r, a', r'\}$
$\mathcal{A}_3 = \{r'''r'ry^2 - r''^2ry^2 + 2r''r'^2y^2 - r'''r'r^3 - 2r''r'^2r^2, r'''r'rx^2 - r''^2rx^2 + 2r''r'^2x^2 + r''^2r^3, -r'a + r'''r + 3r''r', r''''r'r^2 - r'''r''r^2 + 6r'''r'^2r + 6r''r'^3\}$
$\mathcal{A}_4 = \{r'''r'ry^2 - r''^2ry^2 + 2r''r'^2y^2 - r'''r'r^3 - 2r''r'^2r^2, r'''r'rx^2 - r''^2rx^2 + 2r''r'^2x^2 + r''^2r^3, r'a + r'''r + 3r''r', r''''r'r^2 - r'''r''r^2 + 6r'''r'^2r + 6r''r'^3\}$
不难看出, r, a 不是上述方程系统的参变量. 所以, 在升列中出现 r 自己的方程与 r, a 的方程. 此时, 关于 r, a 的方程显然不是唯一的, 也不一定是我们要寻找的方程. 而我们要找的关系 $2ar' + a'r$ 并不在升列中出现. 为了解决这一问题, 我们将在下面引入新的技巧.

4. 代数关系的自动发现.　由例 5.4.8 可以看到, 使用本节所述的方法, 一般发现的关系式是一个或一组关于 $u_1, \cdots, u_q$ 的微分方程. 而我们希望知道这些微分方程解的一般形式. 例如, 在例 5.4.7 中, 给出了 k_1 和 t_1 之间的关系 $k_1't_1'' - t_1'k_1'' = 0$. 现在我们想问, k_1 和 t_1 是否存在代数关系. 考虑下面问题:

问题 A　如果 $x_1, \cdots, x_n$ 满足微分方程 $\mathbb{H}$ 与非等式 $\mathbb{D}$, 确定 x 是否满足关系式 $R(a_1, \cdots, a_m, x_1, \cdots, x_n) = 0$, 其中 R 是有理系数多项式, $a_1, \cdots, a_m$ 为常数.

引理 5.4.9　*变量 $x_1, \cdots, x_n$ 满足常系数 d 次多项式方程的充要条件是*

$$LD_d = \mathrm{wronskian}(1, x_1, \cdots, x_n, x_1^2, x_1x_2, \cdots, x_n^2, \cdots, x_1^d, \cdots, x_n^d) = 0$$

由此引理, 得到问题 A 的下列解法:

1. 利用零点分解算法, 我们可以得到 (变量序并不重要)

$$\mathrm{Zero}(\mathbb{H}/\mathbb{D}) = \cup_{i=1}^{t}\mathrm{Zero}(\mathrm{sat}(\mathcal{A}_i)/\mathbb{D})$$

2. 对于 $k = 1 \to \infty$, 设 k_0 是第一个满足 $\mathrm{prem}(LD_{k_0}, \mathcal{A}_i) = 0 (i = 1, \cdots, t)$ 的整数. 由引理5.4.9, 则 x 满足常系数 k_0 次多项式关系式.
3. 由引理 5.4.9, 我们可以进一步确定出现在多项式方程中的单项式.

我们只给出问题 A 的半判定解法: 如果存在一个代数关系式, 则我们的方法可以在有限步找到该关系式. 但如果该关系式不存在, 这一方法不会停止.

例 5.4.10(考虑例 5.4.7)　*我们求得 k_1 和 t_1 之间的关系式 $h_1 = k_1't_1'' - t_1'k_1'' = 0$. 由上述算法, $LD_1 = \mathrm{wronskain}(1, k_1, t_1) = k_1't_1'' - t_1'k_1'' = h_1$, 则 k_1 和 t_1 对于任意常数 a, b, c 满足线性关系式 $ak_1 + bt_1 + c = 0$.*

例 5.4.11　*设一个质点在一个向心力的作用下在平面上运动, 向心力的大小正比于质点与向心力中心的距离, 求质点的轨迹.*

定义质点在 t 时刻的坐标为 $(x(t), y(t))$. 假设作用力中心在原点 $(0,0)$. 记 a 为质点的加速度. r 为质点到原点之间的距离. 问题可以表述为 $(\mathbb{H}, \mathbb{D})$:

$\mathbb{H} = \{h_1, h_2, h_3, h_4\}$

$h_1 = r^2 - x^2 - y^2 = 0$

$h_2 = a^2 - x''^2 - y''^2 = 0$

$h_3 = x''y - y''x = 0$ 作用力指向原点.

$h_4 = \text{wronskian}(a, r) = a'r - r'a = 0$ a 正比于 r.

$\mathbb{D} = \{a, r, d_1\}$

$d_1 = ld(1, x, y) = x'y'' - y'x'' \neq 0$ 轨迹不是一条直线.

```
>depend([a, r, y, x], [t]);
>dwsolve([r^2-x^2-y^2, a^2-x[2]^2-y[2]^2, x*y[2]-x[2]*y,
a*r[1]-a[1]*r], [a, r, y, x], [a, r, x[1]*y[2]-x[2]*y[1]]);
```

使用上述命令, 我们有

$$\text{Zero}(\mathbb{H}/\mathbb{D}) = \text{Zero}(\text{sat}(\mathcal{A}_1)/\mathbb{D})$$

其中

$$\mathcal{A}_1 = \quad x^2a^2 - x''^2y^2 - x''^2x^2 \quad -r^2 + y^2 + x^2 \quad -x''y + y''x \quad x'''x - x'x''$$

可以用 MMP 的函数 dpremas 验证, 满足条件 $\text{dpremas}(LD_k, \mathcal{A}_1) = 0$ 的最小 k 为 2.

```
>A1:=[x^2*a^2-x[2]^2*y^2-x[2]^2*x^2,-r^2+y^2+x^2,
-x[2]*y+y[2]*x,x[3]*x-x[1]*x[2]];
>dpremas(wronskian([1, x, y, x^2, x*y, y^2], t), A1, [a, r, y, x]);
0
```

因此轨迹是一个圆锥曲线. 更进一步可以验证 $\text{dpremas}(\text{wronskian}(1, x^2, xy, y^2), \mathcal{A}_1) = 0$. 集合 $\{1, x^2, xy, y^2\}$ 中的项不能够再去掉. 因此轨迹是以 $(0,0)$ 为中心的二次曲线: $ax^2 + bxy + cy^2 = 1$, 其中 a, b 和 c 为任意常数.

例 5.4.12 *如果质点的轨迹是一条中心在作用力中心的椭圆. 求作用力大小和质点与作用力中心距离的关系.*

利用例 5.4.11 相同的符号. 假设椭圆的方程为 $ax^2 + by^2 + c = 0$, 其中 a, b 和 c 是常数. 这一关系等价于 $h_5 = \text{wronskain}(1, x^2, y^2) = 0$. 问题变为在 $h_5 = 0 \wedge h_3 = 0 \wedge d_1 = \text{wronskain}(1, x, y) \neq 0$ 的条件下求 a 和 r 之间的关系.

```
>depend([a, r, y, x], [t]);
```

```
>dwsolve([r^2-x^2-y^2, a^2-x[2]^2-y[2]^2, x*y[2]-x[2]*y,
wronskian([1, x^2, y^2], t)],[a, r, y, x],
[x[1]*y[2]-x[2]*y[1]]);
```

使用上述命令, 我们有 $\mathrm{Zero}(\mathbb{H}/\mathbb{D}) = \mathrm{Zero}(\mathrm{sat}(\mathcal{A}_1)/\mathbb{D})$, 其中

$$\mathcal{A}_1 = \quad x^2a^2 - x''^2y^2 - x''^2x^2 \quad -r^2 + y^2 + x^2 \quad x''y - y'x' \quad x'''x - x'x''$$

不难验证 $\mathrm{wronskian}(a, r)$ 对以上升列的余式为零. 说明 a 与 r 成正比.

例 5.4.13 我们现在再由 Kepler 定律自动推导 Newton 引力定律.

在变量顺序 $e < p < x < y < r < a$ 下使用零点分解定理:

```
>depend([a, r, x, y], [t]);
>vs:=[a, r, y, x, p, e];
>ps:=[x* diff(y, t, 2)-diff(x, t, 2)*y,
r^2-x^2-y^2, a^2-x[2]^2-y[2]^2, r - p - e * x];
>dwsolve(ps, vs, [a, p], "weak");
```

我们有 $\mathrm{Zero}(\mathbb{H}/\mathbb{D}) = \cup_{1\leqslant i\leqslant 2}\mathrm{Zero}(\mathrm{sat}(\mathcal{A}_i)/\mathbb{D})$, 其中

$\mathcal{A}_1 = \{e^2x^2a - x^2a + 2epxa + p^2a - px'^2, -r + ex + p, -y^2 + e^2x^2 - x^2$
$+ 2epx + p^2, e^3x''x^3 - ex''x^3 + 3e^2px''x^2 - px''x^2 + 3ep^2x''x + px'^2x + p^3x''\}$

$\mathcal{A}_2 = \{e^2x^2a - x^2a + 2epxa + p^2a + px'^2, -r + ex + p, -y^2 + e^2x^2 - x^2$
$+ 2epx + p^2, e^3x''x^3 - ex''x^3 + 3e^2px''x^2 - px''x^2 + 3ep^2x''x + px'^2x + p^3x''\}$

不难验证在上面每一个分支上 $LD_3(a, r)$ 为零. 说明 a, r 满足一个三次代数关系. 进一步化简可以知道 $\mathrm{wronskian}(ar^2, 1) = 0$, 即得到 Newton 引力定律.

文献说明

MMP 所采用的初等几何机器证明的基本原理由吴文俊提出 [85]. 吴文俊在两本专著中对这一方法有详尽介绍 ([86] 和 [87]). 关于几何命题的各种代数表示请见 [14]. 构造型几何命题非退化的充分性在 [22] 中给出. 几何命题最优非退化条件的生成算法在 [13] 中给出. wprove 基本上采用了 [21] 和 [22] 中改进的吴方法.

微分几何定理的机器证明方法由吴文俊提出 ([94],[88] 和 [91]). 使用 wronskian 引入微分几何语言在 [18] 中提出. 自动发现有微分方程定义的代数关系的算法在 [19] 中提出.

初等与微分几何定理的自动发现的方法由吴文俊提出 ([92]). 几何公式的存在性与唯一性在 [21] 中证明. 完整的几何公式的发现在 [13] 中提出. 关于微分几何定理证明更多的例子请见 [16].

第六章　代数方程求解

方程求解是数学机械化理论研究的核心内容. 方程求解不仅是重要的数学问题, 还在众多的领域存在应用. 实际上, 科学与工程中的很多问题都可以转化为方程求解. 著名思想家 Descartes 曾提出下面解决任意问题的方案：

$$
\begin{aligned}
\text{任何问题的解答} &\Rightarrow \text{数学问题的解答}\\
&\Rightarrow \text{代数问题的解答}\\
&\Rightarrow \text{代数方程组求解}\\
&\Rightarrow \text{单个代数方程求解}
\end{aligned}
$$

由于 Descartes 所处的时代, 这一设想有其明显的局限性. 但是, 正如著名数学家 Polya 所讲, "这一构想虽未成功, 但它仍不失为一个伟大的设想. 即使失败了, 它对于科学发展的影响比起千万个成功的小设想来, 仍然要大的多". 这是因为, 这一设想虽然不能适用于所有问题, 但确实有很多重要问题可以按照这一设想解决. 例如, 本书第五章中介绍的几何定理证明的吴方法实际上就是方程求解的应用. 本章将介绍怎样用 MMP 求解代数方程.

§6.1　多项式方程求解的吴消元法

提到方程组求解, 人们往往会想到数值方法, 包括常用的 Newton 迭代方法, 同伦连续方法等等. 这些方法从方程组给定的某些初始值出发, 通过迭代收敛到给定多项式方程组的某个解. 通常这种方法只能给出所有解的集合中的一个解, 并且是解的近似值. 方程的近似求解对于工程应用当然是非常重要的. 但很多重要的应用往往需要我们知道方程的精确解. 例如, 第五章中的几何定理证明需要对方程的精确解的结构进行分析. 又如, 理论物理中的杨 Baxter 方程的求解, 信息安全对于某些密码的破译, 以及本书 §6.5 中提到的一系列应用, 都要求我们对方程精确求解.

与数值计算相对应, 我们可以采用符号计算方法求解方程. 符号计算方法可以给出任意代数方程组的全部精确解, 在所给方程组为高维情形时给出其全部流形解. 因此, 基于符号计算的方法是一种完全的方程求解方法. 目前主要有三种方程求解的符号计算方法：结式方法、Buchberger 提出的 Gröebner 基方法、与吴文俊提出的特征列方法. MMP 中实现的是特征列方法.

1. 单变量方程的求解. 我们可以用 MMP 的函数 roots求得 *单变量多项式方程* 的精确解. 例如,

```
>p1:=3x^5-92x^4+1016x^3-5074x^2+11413x-9282:
>roots([p1], [x]);
```

$[[x,2],[],\left[x,\frac{17}{3}\right],[],[x,13],[],[x,3],[],[x,7],[]]$

函数 roots(ps, vs) 的第一个参数 ps 是待求解多项式的链表, 第二个参数 vs 是待求解的未知数的链表. 其返回值是一个偶数长度的链表, 其中每个元素仍是链表. 每两个链表表示方程的一组解：奇数位置的子链表存放各个待求解的未知数及其所对应的解, 偶数位置的子链表存放在方程中出现的其他符号参数. 本例中, 待求解的多项式和未知数都只有一个. 由于该方程中不包含其他符号参数, 所以返回值中的第偶数个子链表都是空表.

```
>p2:=x^4-5x^2+6x-2:
>roots([p2], [x]);
```

$[[x,-\sqrt{3}-1],[],[x,\sqrt{3}-1],[],[x,1],[]]$

在这个例子中, 方程 p2=0 有一个二重根 $x=1$. 函数 roots 会把重复的根舍去, 这样做的优点是简化了解的输出形式, 但其缺点是无法知道每个解的重数. 下面我们再来看两个有复数根的多项式方程求解的例子：

```
>p3:=x^3+x-2:
>roots([p3], [x]);
```

$\left[\left[x,\frac{-I\sqrt{7}-1}{2}\right],[],\left[x,\frac{I\sqrt{7}-1}{2}\right],[],[x,1],[]\right]$

roots 找到了这个三次方程的三个解, 包括一个实数解和两个共轭虚数解.

```
>p4:=x^4+5x^2+6x-2:
>roots([p4], [x]);
```

$\left[\left[x,\frac{\sqrt{\sqrt{33}-5}\sqrt{2}}{2}\right],[],\left[x,\frac{-\sqrt{\sqrt{33}-5}\sqrt{2}}{2}\right],[],\left[x,\frac{1}{2I}\sqrt{\sqrt{33}+5}\sqrt{2}\right],[],\right.$
$\left.\left[x,-\frac{1}{2I}\sqrt{\sqrt{33}+5}\sqrt{2}\right],[]\right]$

这是一个四次方程. 读者可以注意到, p4 是对上面 p2 略作改变后得到的多项式. 我们得到了它的四个不同的根, 包括两个实数根和两个共轭虚数根.

在用 roots 求解的方程中除了包含待求解的未知数外, 还可以包含其他的符号参数. 此时 roots 将会把这些符号参数视作自由变元, 并把待求解的未知数用这些符号参数来表示. 作为例子, 下面我们来求解初等代数中最基本的一元二次方程：“$ax^2+bx+c=0$”, 方程中包含三个符号参数 a, b 和 c.

```
>p5:=a*x^2+b*x+c:
>roots([p5], [x]);
```

$\left[\left[x, \dfrac{-b-I\sqrt{4ca-b^2}}{2a}\right], [], \left[x, \dfrac{-b+I\sqrt{4ca-b^2}}{2a}\right], []\right]$

次数小于等于四的方程的解可以用根式严格表达, 但是次数大于四的方程的解则一般不能用根式严格表示. 那么, 怎样精确求解这样的单变量方程呢？此时函数 roots 将以 “RootOf” 的形式给出方程的形式解. 例如：

```
>p6:=3*x^5-92*x^4+1016*x^3-5074*x^2+1413*x-9282:
>roots([p6], [x]);
```

$[[x, \mathrm{RootOf}(3x^5-92x^4+1016x^3-5074x^2+1413x-9282)], []]$

“RootOf(poly)”函数表示不可约多项式方程 poly=0 的任意一个根. 我们之所以可以这样做, 是因为不可约多项式方程的所有根在一定意义下是等价的. 例如, 假设 $p(x)=0$ 是一个不可约多项式方程, $q(x)$ 是另外一个多项式. 如果 $p(x)=0$ 的一个根 x_1 使得 $q(x_1)=0$, 则对 $p(x)=0$ 的任何一个根 x_2 都有 $q(x_2)=0$. 也就是说, 性质 $q(\mathrm{RootOf}(p))=0$ 对于不可约多项式 p 是有意义的. 很明显, $q(\mathrm{RootOf}(p))=0$ 当且仅当 p 是 q 的因子.

我们还可以进一步基于实根隔离算法给出方程实解的唯一的精确表示, 具体请见 §8.1.

2. 多变量方程组的求解. 考虑具有任意变元个数与具有任意方程个数的非线性多项式方程组：

$$\mathbb{P}=\{p_1(x_1,\cdots,x_n)=0,\cdots,p_r(x_1,\cdots,x_n)=0\}$$

其中 $p_i\in\mathbb{K}[\mathbb{X}], \mathbb{X}=(x_1,\cdots,x_n)$, 并且 $\mathbb{K}$ 是特征为 0 的域, 通常为有理数域 $\mathbb{Q}$. 我们将考虑方程组在复数域 $\mathbb{C}$ 上的解. $\mathbb{P}=0$ 的所有解的集合称为由 $\mathbb{P}=0$ 定义的代数簇, 记为 Zero($\mathbb{P}$).

方程组的精确求解的本质与其说是给出解的计算方法, 不如说是给出解的结构. 例如, 对于任意给定的方程组 $\mathbb{P}=0$, 我们希望知道其解的下列基本性质.

- **解的个数.** 如果方程组有有限个解, 则称方程组的解空间是 *零维的*. 如果方程组有无穷个解, 则称方程组的解空间是 *高维的*, 此时解空间的几何维数称为方程组定义的代数簇的 *维数*. 方程组的高维解又称为其 *流形解*. 一个代数簇称为 *齐维的*, 如果其所有不可约分支的维数相同.
- **解的形式.** 如果一个方程组存在解, 我们怎样表示这些解呢? 为此, 我们重新考虑第四章引进的三角列概念. 考虑一个多项式集合 $\mathcal{A}\subset\mathbb{K}[\mathbb{X}]$. 如果 $x_1,\cdots,x_n$ 可以重新排列为 $u_1,\cdots,u_q,y_1,\cdots,y_p$ $(p+q=n)$, 使得 $\mathcal{A}$ 中的多项式 $A_1,\cdots,A_p$

可以被重排为如下形式, 则称 $\mathcal{A}$ 是一个 *三角列*.

$$\begin{aligned} A_1(\mathbb{U}, y_1) &= I_1 * y_1^{d_1} \ + \text{关于 } y_1 \text{ 的低次项} \\ A_2(\mathbb{U}, y_1, y_2) &= I_2 * y_2^{d_2} \ + \text{关于 } y_2 \text{ 的低次项} \\ &\cdots\cdots\cdots\cdots \\ A_p(\mathbb{U}, y_1, \cdots, y_p) &= I_p * y_p^{d_p} \ + \text{关于 } y_p \text{ 的低次项} \end{aligned} \tag{6.1}$$

如果 A_j 对于 $A_i, i < j$ 约化, 则称 $\mathcal{A}$ 为一个 *升列*. 变量 $u_1, \cdots, u_q$ 称为 $\mathcal{A}$ 的 *参变量*. 参变量的个数称为 $\mathcal{A}$ 的维数, 记为 $\dim(\mathcal{A})$. 一个三角列的解空间是基本确定的. 例如, 给定 u_i 的一组值, 如果所有的初式 I_i 都不为零, 我们一般可以由 $A_1 = 0, A_2 = 0, \cdots, A_p = 0$ 顺序求解 $y_1, y_2, \cdots, y_p$, 而且解的个数一般是 $\prod_{i=1}^{p} d_i$. 换言之, $\mathrm{Zero}(\mathcal{A}/J)$ 可以被认为是一个可确定的集合. 一般地, 我们有下面结果：

定理 6.1.1　*设 $\mathcal{A}$ 是任意升列, J 是 $\mathcal{A}$ 的初式的乘积. 则有*

- $\mathrm{Zero}(\mathcal{A}/J)$ *与* $\mathrm{Zero}(\mathrm{sat}(\mathcal{A}))$ *或者是空集或者是齐维的, 且其分支的维数就是* $\mathcal{A}$ *的维数.*
- *如果* $\mathcal{A}$ *是不可约升列, 则* $\mathrm{Zero}(\mathrm{sat}(\mathcal{A}))$ *是* $\dim(\mathcal{A})$ *维的不可约代数簇.*

既然三角列的解在一定意义下已经确定, 对于一般形式的方程组, 我们只需要将其解空间分解为三角列的解, 就解决了方程组的求解问题. 如第四章所述, 这一分解可以由零点分解定理得到. 重述如下.

定理 6.1.2 (零点分解定理)　*对于给定的多项式集合 $\mathbb{P}$, 存在算法确定一个有限的升列集合 (或三角列集合) $\mathcal{A}_j$ 使得*

$$\mathrm{Zero}(\mathbb{P}) = \cup_j \mathrm{Zero}(\mathcal{A}_j/\mathrm{I}_{\mathcal{A}_j}) = \cup_j \mathrm{Zero}(\mathrm{sat}(\mathcal{A}_j)) \tag{6.2}$$

MMP 的函数 roots 实现了上述方程组求解的吴方法. roots 主要执行下面两个步骤：

- 调用吴零点分解定理将一般形式方程组的解分解为三角形式方程组的解.
- 对三角列中的方程单独求解. 如果某些方程不能精确求解, 则用 RootOf 函数表示.

下面我们来看几个求解多项式方程组的例子.

```
>p1:=x^2+y^2-41:
>p2:=x+y-9:
>roots([p1, p2], [x, y]);
```

$[[y, 4, x, 5], [], [y, 5, x, 4], []]$

在上一个小节, 我们已经介绍过 roots 的输入参数和返回值形式, 这里不再重复. 在本例中, 我们通过 roots 得到了方程组的两组精确解. 以上求解过程可以分为两步：首先将方程组化为三角列形式.

```
>wsolve([p1, p2], [x, y], [], "ritt");
```

我们得到两个分支：

$$[[x-5, y-4], [x-4, y-5]]$$

然后对每个分支分别求解即可得到结论. 在 roots 也可以求解带有符号参数的方程组. 我们来看下面例子：

```
>p1:=x^2+(p-10)*x-9*p+9:
>p2:=y^3-x+1:
>roots([p1, p2], [x, y, p]);
```

$$\Big[[y, -I\sqrt{3}-1, x, 9], [p], [y, I\sqrt{3}-1, x, 9], [p], [y, 2, x, 9], [p], [y, -\sqrt[3]{p}, x, -p+1], [p],$$

$$\left[y, \frac{1}{2}I\sqrt[3]{p}\sqrt{3}+\frac{1}{2}\sqrt[3]{p}, x, -p+1\right], [p], \left[y, -\frac{1}{2}I\sqrt[3]{p}\sqrt{3}+\frac{1}{2}\sqrt[3]{p}, x, -p+1\right], [p]\Big]$$

该方程组包含一个符号参数 p, 方程求解时它将被视作自由变元, 出现在方程解的表达式中. 我们利用 roots 获得了该方程组的六组解, 并且每一组解都是以显式表达式的方式给出的. 我们看到, 符号参数 p 存放在每个偶数位置上的子链表中.

方程求解是中国古代数学研究的中心问题. 早在公元 1303 年元朝朱世杰的著作《四元玉鉴》中已经出现了一种求解一般非线性方程组的思想. 下面的方程组是朱世杰考虑的一个具体例子.

```
>p1:=x*y*z-x*y^2-z-x-y;
>p2:=x*z-x^2-z-y+x;
>p3:=z^2-x^2-y^2;
>roots([p1, p2, p3], [x, y, z]);
```

输入上述命令得到方程组的六组解：

$$\Big[[y, -z, x, 0], [z], \left[z, -\frac{1}{2}I\sqrt{2}, y, \frac{1}{2}I\sqrt{2}, x, 0\right], [], \left[z, \frac{1}{2}I\sqrt{2}, y, -\frac{1}{2}I\sqrt{2}, x, 0\right], [],$$

$$[z, -1, y, 0, x, 1], [], [z, 5, y, 4, x, 3], []\Big]$$

其中第一组解 $y=-z, x=0$ 是方程组的流形解. 与这组解相对应的三角列是 $[x, y+z]$. 其他解是零维解. 不妨再看看其相应的三角列：

```
>wsolve([p1, p2, p3], [x, y, z], [], "ritt");
```

得到以下分解：

$$[[x, y+z], [x, y+z, 2*z^2+1], [x-1, y, z+1], [x-3, y-4, z-5], [x, y, z]].$$

考虑下面方程组：

```
>p1:=x*y*z-x^2-y^2-z+1;
>p2:=x*z+z^2-1;
>p3:=z^2+x^2+y^2-2;
>roots([p1, p2, p3], [x, y, z]);
```

得到其解如下：

$$\Big[[z, \mathrm{RootOf}(2z^8-7z^6-2z^5+10z^4+2z^3-5z^2+1), y, \frac{(z^2-z-1)}{(z^2-1)}, x, \frac{(-z^2+1)}{(z)}], []\Big].$$

这里, 我们使用了 RootOf 函数. 之所以可能是因为三角列的下面性质：

定理 6.1.3 对于不可约升列 $\mathcal{A}$ 与多项式 P, 我们有 P 在 $\mathrm{Zero}(\mathrm{sat}(\mathcal{A}))$ 上为零当且仅当 $\mathrm{prem}(P, \mathcal{A}) = 0$.

利用上述判定准则, 我们很容易知道一个多项式是否在一个不可约升列的零点集合上为零. 例如在上例中, 我们可以判定 z^2-1 在 $2z^8-7z^6-2z^5+10z^4+2z^3-5z^2+1$ 的零点上非零, 所以, 表达式 $y = (z^2-z-1)/(z^2-1)$ 是有意义的.

§6.2 预解式及其应用

1. 预解式. 通过将一般形式的方程组化为三角形式, 我们已经将方程组的求解化为单变量、单个方程的求解. 但是, 为了求解三角形式的方程组, 我们需要连续求解多个非线性方程. 在这一过程中, 由于后面方程的求解依赖于前面方程的解, 而这些解又是近似的. 所以, 在这个求解过程中可能会造成误差的积累. 解决这个问题的一个途径是预解式理论. 通过使用预解式, 我们可以将一个方程组的解直接化为单个非线性方程的求解. 其他所有变量的值可以通过计算非线性方程的解的有理函数给出. 具体, 我们有

定理 6.2.1 设 $\mathcal{A}$ 是一个形如 (6.1) 的不可约升列. 我们可以求得整数 $c_1, \cdots, c_p$ 与新变元 $w = c_1y_1 + \cdots + c_py_p$, 使得在变元顺序 $u_i < w < y_1 < \cdots < y_p$ 下有

$$\mathrm{Zero}(\mathrm{sat}(\mathcal{A}) \cup \{w - c_1y_1 - \cdots - c_py_p\}/J) = \mathrm{Zero}(\mathrm{sat}(\mathcal{A}')/J)$$

其中 J 是 $\mathcal{A}$ 的初式的乘积. $\mathcal{A}'$ 具有下面形式

$$\begin{aligned}
&R(u_1, \cdots, u_q, w)\\
&I_1(u_1, \cdots, u_q, w)y_1 - U_1(u_1, \cdots, u_q, w)\\
&I_2(u_1, \cdots, u_q, w)y_2 - U_2(u_1, \cdots, u_q, w)\\
&\qquad\qquad\cdots\cdots\cdots\cdots\\
&I_p(u_1, \cdots, u_q, w)y_p - U_p(u_1, \cdots, u_q, w)
\end{aligned}$$

其中 R, I_i, U_i 均为变量 u_i 与 w 的多项式. 我们称 R 是素理想 $\mathrm{sat}(\mathcal{A})$ 的预解式. 由此得到不可约代数簇的预解式表示：

$$R(u_1, \cdots, u_q, w) = 0, y_1 = \frac{U_1}{I_1}, y_2 = \frac{U_2}{I_2}, \cdots, y_p = \frac{U_p}{I_p}$$

关于整数 c_i 的选取, 我们有下面结论：随机选取 c_i 为整数或有理数, 则定理 6.2.1 成立的概率为 1. 我们看下面的例子：

```
>p1:=x*y+y^2-3;
>p2:=z^2+x^2+y-2;
>p3:=w-y-z;
>wsolve([p1, p2, p3], [z, y, w, x],[],"wu");
```

得到方程组 p1=0, p2=0 的预解式表示：

$$\begin{aligned}
R = {} & w^4 + 2*x*w^3 + (3*x^2 - x - 10)*w^2 + (2*x^3 - 10*x + 12)*w + \\
& 2*x^4 - x^3 + 5*x - 2 \\
z = {} & (w^2 + x*w - w - x^2 - 1)/(2*w + x - 1) \\
y = {} & (w^2 + x^2 + 1)/(2*w + x - 1)
\end{aligned}$$

由上例可以看到, 使用预解式只是在理论上简化了方程求解过程. 由于新变量的引进, 使得方程组变为“稠密形式”, 从而增加了计算量. 另一方面, 预解式在理论上是很强大的, 很多问题的求解可以归结为预解式的求解. 例如第四章中的因式分解算法 algfactor 就使用了这一方法. 下面举例说明预解式的其他应用.

2. 不可约代数曲线的有理参数化. 代数几何中有以下定理：

定理 6.2.2 任意 r 维不可约代数簇与 $\mathbb{E}^{r+1}$ 空间中某个超曲面双有理等价.

我们可以证明以下更一般的结果.

定理 6.2.3 设 $\mathbb{P}$ 是 $\mathbb{K}[\mathbb{U}, \mathbb{Y}]$ 中的有限多项式集合, $\mathbb{U}$ 是理想 $(\mathbb{P})$ 的参数集合, 即：理想 $(\mathbb{P}) \cap \mathbb{K}[\mathbb{U}] = \{0\}$ 且对 $y_i \in \mathbb{Y}$ $(\mathbb{P}) \cap \mathbb{K}[\mathbb{U}, y_i] \neq \{0\}$. 则存在 w 与 $\mathbb{U}$ 的多项式 R 使得 $\mathrm{Zero}(\mathbb{P})$ 与 $\mathrm{Zero}(R)$ 双有理等价. 同时可以找到对应的双有理映射.

证明 我们可以求得整数 $M_1, \cdots, M_p$, 使得 $RD = \mathrm{Radical}(\mathbb{P} \cup \{w - (M_1 y_1 + \cdots + M_p y_p)\})$ 的特征列具有下述形式

$$R(u, w), R_1(u, w, y_1), \cdots, R_p(u, w, y_p) \tag{6.3}$$

其中 $R_i = I_i y_i - U_i$. $R = 0$ 是 $(\mathbb{P})$ 的一个预解式. 定义映射

$$MP_1 : \mathrm{Zero}(\mathbb{P}) \to \mathrm{Zero}(R)$$

$MP_1(u_1, \cdots, u_q, y_1, \cdots, y_p) = (u_1, \cdots, u_q, M_1y_1 + \cdots + M_py_p)$. 定义另一个映射

$$MP_2 : \text{Zero}(R) \to \text{Zero}(\mathbb{P})$$

为 $MP_2(u_1, \cdots, u_q, w) = (u_1, \cdots, u_q, U_1/I_1, \cdots, U_p/I_p)$. 令 $I = \prod_{i=1}^{p} I_i$. 则 MP_2 在 $D_1 = \text{Zero}(R) - \text{Zero}(I)$ 上完全确定. 由于 $\text{Radical}(\mathbb{P})$ 不包含非零的 $\mathbb{U}$ 多项式, $\text{Radical}(\mathbb{P}) = \cap\mathbb{P}_i$, 其中 $\mathbb{P}_i$ 是参数集合 $\mathbb{U}$ 的素理想. 所以, 多项式 R 无只包含 u 的因子. 因此除了 $\text{Zero}(I)$ 中维数低于 q 的一部分外, MP_2 在 $\text{Zero}(R)$ 均有定义. 我们可以确认 $MP_1(MP_2)$ 和 $MP_2(MP_1)$ 为恒同映射. 因此, $\text{Zero}(R)$ 和 $\text{Zero}(\mathbb{P})$ 为双有理的. 它们之间的映射为 MP_1 和 MP_2.

不可约代数曲线是一维不可约代数簇. 设 $C = \text{Zero}(\mathbb{P})$ 为一个不可约的代数曲线, 其中 $\mathbb{P} \subset \mathbb{K}[\mathbb{X}]$. 如果存在不定元 t 的多项式 $u_1, \cdots, u_n, w$ 满足 $\gcd(u_1, \cdots, u_n, w)$ 为 $\mathbb{K}$ 中的非零元, 且 $\forall P \in \mathbb{P}$, $P(u_1/w, \cdots, u_n/w) \equiv 0$. 则称 C 为 *有理的*. 称

$$x_1 = u_1/w, \cdots, x_n = u_n/w$$

是该曲线的*参数表示*. u_i 和 w 的最大次数称为 *参数方程组的次数*.

下面我们给出确定一个代数曲线是否为有理的判别方法. 若为有理的话, 该方法同时给出它的一个参数方程组.

定理 6.2.4　*对于 A^n 中的不可约代数曲线 $C = \text{Zero}(\mathbb{P})$, 我们可以求得不可约多项式 $f(x, y)$, 使得 C 与 $\text{Zero}(f)$ 是双有理等价的. 我们可以同时给出 C 和 $\text{Zero}(f)$ 之间的双有理映射.*

素理想的维数等于其参数的数目. 不可约代数曲线 C 有一个参数, 因此 C 的一个预解式是一个双变量的多项式. 由定理 6.2.3 我们可以求得一个平面代数曲线与 C 双有理等价.

显然, C 是有理的充要条件是 $f(x, y) = 0$ 是有理的. 更进一步, 利用 C 和 $f = 0$ 之间的双有理映射, 由 C (或 $f = 0$) 可以给出 $f = 0$ (或 C) 的参数方程. 因此, 求 C 的有理参数方程组问题变为 $f(x, y) = 0$ 的有理参数化问题. 我们可以首先计算 $f(x, y) = 0$ 的亏格. 如果该亏格为零, 则 f 是有理的. 如果该亏格非零, f 不是有理的.

曲线 C 的参数方程组 $x_i = u_i/w$ 称为 *正则* 的, 如果除有限点外, C 上的任一点 $(x'_1, \cdots, x'_n)$ 仅存在唯一的 t_0 满足 $x'_i = u_i(t_0)/w(t_0), i = 1, \cdots, n$. 由 Lüroth 定理, 有理曲线总存在一个正则的参数方程组.

定理 6.2.5　*令 $x = u(t)/w(t), y = v(t)/w(t)$ 是平面曲线 $f(x, y) = 0$ 的一个正则参数方程组, 假设 $\gcd(u, v, w) = 1$, 则 f 的次数等于参数方程组的次数.*

由此可以得到下面计算代数曲线有理参数方程组的算法. 令 $\mathbb{P}$ 是 $\mathbb{K}[\mathbb{X}]$ 上的一个多项式的有限集合. 下述算法确定 $C = \mathrm{Zero}(\mathbb{P})$ 是否为有理不可约代数曲线. 若是, 则给出 C 的一个参数方程组.

1. 由吴零点分解定理, 在变量序 $x_1 < x_2 <, \cdots, < x_n < t$ 下, 我们可以得到如下不可约分解：

$$\mathrm{Zero}(\mathbb{P}) = \cup_{i=1}^{m} \mathrm{Zero}(\mathrm{sat}(\mathcal{A}_i))$$

其中 $\mathcal{A}_i$ 是一个不可约升列. C 是不可约代数曲线的充要条件是 $m = 1$ 且 $\mathcal{A}_1$ 包含 $n-1$ 个多项式. 如果 C 是不可约曲线, 转向 2. 否则, 算法停止.

2. 重新命名变量得到

$$\mathcal{A}_1 = A_1(x, y_1), \cdots, A_p(x, y_1, \cdots, y_p), p = n-1$$

3. 由定理 6.2.3, 可以找到 $\mathrm{sat}(\mathcal{A}_1)$ 的次数为 d 的预解式 $f(x,y) = 0$, 以及 $f = 0$ 和 $\mathrm{Zero}(PD(\mathcal{A}_1))$ 之间的双有理变换.

4. 令 $d = \mathrm{tdeg}(f(x,y))$ 为 $f(x,y)$ 的全次数.

$$x = u(t)/w(t), y = v(t)/w(t) \tag{6.4}$$

其中 $u(t) = u_d t^d + \cdots + u_0$, $v(t) = v_d t^d + \cdots + v_0$ 和 $w(t) = w_d t^d + \cdots + w_0$ 是关于中间变量 u_i, v_i 和 w_i 的多项式.

5. 分别将 $f(x,y) = 0$ 中的 x 和 y 替换为 $u(t)/w(t)$ 和 $v(t)/w(t)$, 去掉其中分母, 得到 t 的多项式 Q, 其系数为 u_i, v_i 和 w_i 的多项式. 令 t 的多项式 Q 的系数集合为多项式集合 $\mathbb{H} = \{P_1, \cdots, P_h\}$.

6. (6.4) 是 $f = 0$ 的一个参数方程组的充要条件是 $\mathbb{H}$ 具有这样的零点, 当 u, v 和 w 的系数被替换为这些零点时, $u(t)/w(t)$ 和 $v(t)/w(t)$ 不是 $\mathbb{K}$ 中的元素.

7. 令 $\mathbb{D}_1 = \{u_i w_j - u_j w_i \mid i, j = 1, \cdots, d\}$, $\mathbb{D}_2 = \{v_i w_j - v_j w_i \mid i, j = 1, \cdots, d\}$. 则 $f = 0$ 是有理的充要条件是 $HD = \mathrm{Zero}(\mathbb{H}) - (\mathrm{Zero}(\mathbb{D}_1) \cup \mathrm{Zero}(\mathbb{D}_2))$ 非空. 当该条件满足时, HD 的任一零点给出 $f = 0$ 的一个参数方程组.

3. 代数扩域的本原元的计算. 代数扩域理论的一个基本结果是, 特征为零的数域的每一个有限代数扩域中, 存在一个本原元素. 即

定理 6.2.6 令 $\eta_1, \cdots, \eta_m$ 在 K 上是代数的. 则存在 $f_i \in K$, $i = 1, \cdots, m$, 使得 $K(\zeta) = K(\eta_1, \cdots, \eta_m)$, $\zeta = \sum_{i=1}^{m} f_i \eta_i$.

考虑更为一般的问题：

定理 6.2.7 令 η_1 在 K 上代数, 且对 $i = 2, \cdots, m$, η_i 在 $K_{i-1} = K(\eta_1, \cdots, \eta_{i-1})$ 上代数. 则可以找到整数 f_i, $i = 1, \cdots, m$, 使得若 $\zeta = \sum_{i=1}^{m} f_i \eta_i$ 有 $K(\zeta) = K_m$.

假设 η_i 由以下多项式序列给出

$$A_1(x_1)=0, A_2(x_1,x_2)=0,\cdots,A_m(x_1,\cdots,x_m)=0$$

我们有 $A_i(\eta_1,\cdots,\eta_i)=0$, $i=1,\cdots,m$. 不失一般性, 假设每一个 A_i 的初式是 $\mathbb{K}$ 中的非零数. $ID=\mathrm{Ideal}(A_1,\cdots,A_m)$ 定义一个零维代数簇, 即 ID 的参数集合为空集. 则由定理 6.2.1, 可以找到整数 f_i, $i=1,\cdots,m$, 使得在变量序 $w<x_1<\cdots<x_m$ 下 $\mathrm{Radical}(ID, w-\sum_{i=1}^m f_ix_i)$ 的一个特征列具有形式

$$R(w), R_1(w,x_1),\cdots,R_m(w,x_m)$$

其中 $R_i=x_i-U_i(w)$. 将 x_i 替换为 R_i 中的 η_i, 有 $\eta_i=U(\zeta)$, 即 $K(\zeta)=K_m$.

作为例子, 下面计算 $\sqrt{2}$ 与 $\sqrt[3]{3}$ 的本原元.

```
>p1:=x1^2-2;
>p2:=x2^3-3;
>p3:=w-x1-x2;
>wsolve([p1, p2, p3], [x2, x1, w],[],"wu");
```

$$[[w-x1-x2, w^3-3*x1*w^2+6*w-2*x1-3, w^6-6*w^4-6*w^3+12*w^2-36*w+1]]$$

由计算结果得知 $\sqrt{2}+\sqrt[3]{3}$ 是 $\sqrt{2}$ 与 $\sqrt[3]{3}$ 的本原元.

§6.3 含参数方程组的求解

1. 含参数的代数方程组的求解. 考虑以下代数方程组, 希望利用参数 a_1, a_2, a_3, a_4 表示 x_1, x_2, x_3, x_4.

$$\begin{aligned}
&x_4-a_4+a_2=0\\
&x_4+x_3+x_2+x_1-a_4-a_3-a_1=0\\
&x_3x_4+x_1x_4+x_2x_3+x_1x_3+(-a_3-a_1)a_4-a_1a_3=0\\
&x_1x_3x_4-a_1a_3a_4=0
\end{aligned}\tag{6.5}$$

在求解代数方程组的文献中, 这一类参数方程组常在 $\mathbb{A}=\mathbb{B}[x_1,\, x_2,\, x_3,\, x_4]$ 中求解, 其中 $\mathbb{B}=\mathbb{Q}(a_1,a_2,a_3,a_4)$ 是 a_i 的有理函数域. 在 $\mathbb{A}$ 中 (6.5) 的解可以如下给出：

$$x_1^3-\Big(\frac{a_4a_3+a_4a_1+a_1a_3}{a_4-a_2}\Big)x_1^2+\Big(\frac{a_1a_3a_4(a_1+3+a_4)}{(a_4-a_2)^2}\Big)x_1-\frac{a_1^2a_3^2a_4^2}{(a_4-a_2)^3}=0$$

$$\begin{aligned}
&x_2+\frac{a_4^2-2a_2a_4+a_2^2}{a_1a_3a_4}x_1^2+\\
&\quad\frac{(-a_3-a_1)a_4^2+(a_2a_3+a_1a_2)a_4+a_1a_2a_3}{a_1a_3a_4}x_1+a_4-a_2=0
\end{aligned}$$

$$x_3 + \frac{-a_4^2 + 2a_2a_4 - a_2^2}{a_1a_3a_4}x_1^2 +$$
$$\frac{(a_3+a_1)a_4^2 + ((-a_2+a_1)a_3 - a_1a_2)a_4 - a_1a_2a_3}{a_1a_3a_4}x_1 - a_4 - a_3 - a_1 = 0$$
$$x_4 - a_4 + a_2 = 0$$

但上式仅给出了 (6.5) 的一般解, 遗漏了一些特解. 例如 $a_1 = 0, x_1 = 0, x_2 = a_2$, $x_3 = a_3$, $x_4 = a_4 - a_2$ 给出的解不包含在以上的表达式中.

为了考虑一般情形, 引入以下符号. 令 $\mathbb{B} = \mathbb{K}[u_1, \cdots, u_m] = \mathbb{K}[\mathbb{U}]$, $\mathbb{A} = \mathbb{B}[x_1, \cdots, x_n] = \mathbb{B}[\mathbb{X}]$, 其中 $\mathbb{U} = \{u_1, \cdots, u_m\}$, $\mathbb{X} = \{x_1, \cdots, x_n\}$. $\mathbb{B}$ 中的多项式称为一个 $\mathbb{U}$ *多项式*. 对于 $\mathbb{K}[\mathbb{U}, \mathbb{X}]$ 上的多项式集合 $\mathbb{P} = \{p_1, \cdots, p_t\}$ 和 $\mathbb{D} = \{d_1, \cdots, d_r\}$, 考虑下列参数方程系统

$$p_1 = 0 \wedge \cdots \wedge p_t = 0 \wedge d_1 \neq 0 \wedge \cdots \wedge d_r \neq 0 \tag{6.6}$$

或等价地, 考虑 $\mathrm{Zero}(\mathbb{P}/\mathbb{D})$.

假定我们考虑 $\mathbb{K}$ 的扩域 $\mathbb{E}$ 上的解. 对于上述参数方程, 我们可以提出以下问题:

- 对于哪些参数值, 多项式方程 $\mathbb{P} = 0, \mathbb{D} \neq 0$ 存在解?
- 怎样表示这些解?
- 怎样划分参数空间, 使得参数方程在同一范围内具有相同的解的个数. 由此, 我们可以求出参数方程所能取到的所有可能的解的个数.

为了解决上述问题, 引入下列定义. 参数方程组 (6.6) 的 *解函数* 是 $(S, \mathcal{A})$, 其中 S 是 $\mathbb{E}^m$ 中的纯维数拟代数簇, $\mathcal{A}$ 是 $\mathbb{B}[\mathbb{X}] - \mathbb{B}$ 上的三角列, 满足:

- 对每一个 $u' \in S$, 将 $\mathcal{A}$ 的 u 替换为 u' 得到 $\mathcal{A}'$, 则 $\mathrm{Zero}(\mathcal{A}'/J')$ 是 $\mathbb{E}^n$ 中的一个维数为 $DIM(\mathcal{A}) = n - |\mathcal{A}|$ 的纯拟代数簇, 其中 J' 是 $\mathcal{A}$ 的初式中 u 替换为 u' 所得的结果.
- 对任一 $x' \in \mathrm{Zero}(\mathcal{A}'/J')$, $(u', x') \in \mathrm{Zero}(\mathbb{P}/\mathbb{D})$. 称 (u', x') 为 $(S, \mathcal{A})$ 的一个解. 称 $\mathrm{Zero}(\mathcal{A}'/J')$ 的维数为 $(S, \mathcal{A})$ 的维数.

参数方程组 (6.6) 的一个 *覆盖* 是解函数的集合:

$$C = \{(S_1, \mathcal{A}_1), \cdots, (S_s, \mathcal{A}_s)\} \tag{6.7}$$

使得, 任一 $(u', x') \in \mathrm{Zero}(\mathbb{P}/\mathbb{D})$ 是某一个 $(S_i, \mathcal{A}_i)$ 的解.

假设 $C = \{(S_1, \mathcal{A}_1), \cdots, (S_s, \mathcal{A}_s)\}$ 是 (6.6) 的一个覆盖. 我们可以如下解决关于参数方程的前两个问题.

- $\cup_{i=1}^{s} S_i$ 是参数系统 (6.6) 对于 $\mathbb{X}$ 有解的参数的值. 等价地, 我们有

$$\mathrm{Proj}_{\mathbb{X}}(\mathrm{Zero}(\mathbb{P}/\mathbb{D})) = \cup_{i=1}^{s} S_i$$

- $\dim(\mathcal{A}_i)$, $i = 1, \cdots, s$ 是参数系统 (6.6) 的所有可能的维数.
- $\dim(\mathcal{A}_i) = 0$ 的分支 $(S_i, \mathcal{A}_i)$ 给出 (6.6) 的孤立解.
- 设 $M \subset C$ 是满足 $\dim(S_i) = m$ 的解函数 $(S_i, \mathcal{A}_i)$ 的集合, 则在 $\mathbb{K}(\mathbb{U})[\mathbb{X}]$ 中, 有 $\mathrm{Zero}(\mathbb{P}/\mathbb{D}) = \cup_{(S_j,\mathcal{A}_j)\in M} \mathrm{Zero}(\mathcal{A}_j/J_j \cup \mathbb{D})$.

下面给出一个求参数方程组 (6.6) 的覆盖的算法. 这一算法主要有三步：

- 调用吴特征列方法将一般形式的方程组分解为三角形式.
- 对三角列进行投影运算可以转化为对三角列中的单个多项式进行投影运算. 而且, 关于参数的三角列不会被上述运算破坏.
- 求得各个三角列的投影后, 就可以很容易地给出相应的解函数.

上述算法由函数 parasolve(ps, ds, vars, pars)实现, 其中 ps 与 ds 分别是参数方程中等式与非等式型方程, vars 是要求解的变量列表, pars 是参数变量列表.

例 6.3.1 *考虑方程组*：$\mathbb{P} = \{y^2 - zxy + x^2 + z - 1, xy + z^2 - 1, y^2 + x^2 + z^2 - r^2\}$, *其中 r 是一个参数变量.*

在变量序 $r < z < x < y$ 下, 我们有 $\mathrm{Zero}(\mathbb{P}) = \mathrm{Zero}(\mathcal{A}_1/x) \cup \mathrm{Zero}(\mathcal{A}_2)$, 其中

$$\begin{aligned}\mathcal{A}_1 &= \{z^3 - z^2 + r^2 - 1, x^4 + (z^2 - r^2)x^2 + z^4 - 2z^2 + 1, xy + z^2 - 1\}\\ \mathcal{A}_2 &= \{r^4 - 4r^2 + 3, z + r^2 - 2, x, y^2 - r^2 + 1\}\end{aligned}$$

对于 $\mathcal{A}_1$, 我们有 $\mathrm{Proj}_{x,z,y}\mathrm{Zero}(\mathcal{A}_1/x) = \mathbb{E}$. 对于 $\mathcal{A}_2$, $\mathrm{Proj}_{x,z,y}\mathrm{Zero}(\mathcal{A}_2) = \mathrm{Zero}(r^4 - 4r^2 + 3)$. 因此, $\mathrm{Zero}(\mathbb{P})$ 的一个覆盖是

$$\{(\mathbb{E}, \mathcal{A}_1), (\mathrm{Zero}(r^4 - 4r^2 + 3), \mathcal{A}_2')\}$$

其中 $\mathcal{A}_2' = \{z + r^2 - 2, x, y^2 - r^2 + 1\}$. 由于 $r^4 - 4r^2 + 3 = (r^2 - 1)(r^2 - 3)$. 我们也可以得到另一个覆盖

$$\{(\mathbb{E}, \mathcal{A}_1), (\{\pm 1\}, \{z - 1, x, y\}), (\{\pm\sqrt{3}\}, \{z + 1, x, y^2 - 2\})\}$$

在 MMP 中, 上述覆盖可以如下得到.

```
>ps:=[y^2-z*x*y+x^2+z-1, x*y+z^2-1, y^2+x^2+z^2-r^2];
>ds:=[];
>vars:=[y, x, z];
>pars:=[r];
>parasolve(ps, ds, vars, pars);
```

例 6.3.2 方程 (6.5) 来源于求解化学系统的平衡点.

在 $\mathbb{Q}[a_1, \cdots, x_4]$ 中, 有

$$\text{Zero}((6.5)) = \cup_{i=1}^{9} \text{Zero}(\mathcal{A}_i/J_i)$$

其中

$\mathcal{A}_1 = \{(a_4 - a_2)x_1 - a_1a_3, (a_4 - a_2)x_2 + a_4^2 + (-a_3 - 2a_2 - a_1)a_4 + (a_2 + a_1)a_3 + a_2^2 + a_1a_2, x_3 - a_4, x_4 - a_4 + a_2\}$

$\mathcal{A}_2 = \{(a_4 - a_2)x_1 - a_1a_4, (a_4 - a_2)x_2 - a_2a_4 + a_2^2 + a_1a_2, x_3 - a_3, x_4 - a_4 + a_2\}$

$\mathcal{A}_3 = \{(a_4 - a_2)x_1 - a_3a_4, (a_4 - a_2)x_2 - a_2a_4 + a_2a_3 + a_2^2, x_3 - a_1, x_4 - a_4 + a_2\}$

$\mathcal{A}_4 = \{a_3, a_4 - a_2, x_2 + x_1 - a_2, x_3 - a_1, x_4\}$

$\mathcal{A}_5 = \{a_3, a_4 - a_2, x_2 + x_1 - a_1, x_3 - a_2, x_4\}$

$\mathcal{A}_6 = \{a_1, a_4 - a_2, x_2 + x_1 - a_2, x_3 - a_3, x_4\}$

$\mathcal{A}_7 = \{a_1, a_4 - a_2, x_2 + x_1 - a_3, x_3 - a_2, x_4\}$

$\mathcal{A}_8 = \{a_2, a_4, x_2 + x_1 - a_1, x_3 - a_3, x_4\}$

$\mathcal{A}_9 = \{a_2, a_4, x_2 + x_1 - a_3, x_3 - a_1, x_4\}$

由此容易得其覆盖. 在 MMP 中, 其覆盖可以如下得到.

```
>ps:=[x4-a4+a2, x4+x3+x2+x1-a4-a3-a1,
 x3*x4+x1*x4+x2*x3+x1*x3+(-a3-a1)*a4-a1*a3, x1*x3*x4-a1*a3*a4];
>parasolve(ps, [], [x4, x3, x2, x1], [a4, a3, a2, a1]);
```

例 6.3.3 求解下述 Loterra-Volterra 系统的平衡点:

$$x_1' = x_1(1 - x_1 - ax_2 - bx_3), x_2' = x_2(1 - bx_1 - x_2 - ax_3), x_3' = x_3(1 - ax_1 - bx_2 - x_3)$$

注意到对于线性情形, 吴零点分解算法可以自然给出一个覆盖. 对这一例子, 令

$$\mathbb{P} = \{1 - x_1 - ax_2 - bx_3, 1 - bx_1 - x_2 - ax_3, 1 - ax_1 - bx_2 - x_3\}$$

我们有 $\text{Zero}(\mathbb{P}) = \cup_{i=1}^{4} \text{Zero}(\mathcal{A}_i/J_i)$, 其中

$\mathcal{A}_1 = \{(b + a + 1)x_1 - 1, (b + a + 1)x_2 - 1, (b + a + 1)x_3 - 1\}$

$\mathcal{A}_2 = \{b^2 + (-a - 1)b + a^2 - a + 1, (ab - 1)x_2 + (-b + a^2)x_1 - a + 1, (ab - 1)x_3 + ((a + 1)b - a^2 - 1)x_1 - b + 1\}$

$\mathcal{A}_3 = \{ab - 1, (a^2 + a + 1)x_1 - a, (a^2 + a + 1)x_2 - a, (a^2 + a + 1)x_3 - a\}$

$\mathcal{A}_4 = \{a - 1, b - 1, x_3 + x_2 + x_1 - 1\}$

由此容易得到方程组的一个覆盖. 在 MMP 中, 其覆盖可以如下得到.

```
>ps:= [1-x1-a*x2-b*x3, 1-b*x1-x2-a*x3, 1-a*x1-b*x2-x3];
>parasolve(ps, [], [x3, x2, x1], [b, a]);
```

例 6.3.4　考虑来自神经网络的代数方程系统：$\mathbb{P}=\{zx^2+zy^2-cz+1, yx^2+(z^2-c)y+1, (y^2+z^2-c)x+1\}$, 其中 c 是参数.

在 $\mathbb{Q}[c,z,y,x]$ 中, 我们有 $\mathrm{Zero}(\mathbb{P})=\cup_{i=1}^{5}\mathrm{Zero}(\mathcal{A}_i/J_i)$, 其中

$\mathcal{A}_1=\{2cz^4-2z^3-c^2z^2-2cz-1,$
$\quad 2y-2c^2z^3+4cz^2+(c^3-2)z+c^2, 2x-2c^2z^3+4cz^2+(c^3-2)z+c^2\};$

$\mathcal{A}_2=\{2z^3-cz+1, y-z, x-z\};$

$\mathcal{A}_3=\{z^3-cz-1, y^2+zy+z^2-c, x+y+z\};$

$\mathcal{A}_4=\{2z^4-3cz^2+z+c^2, cy-2z^3+2cz-1, x-z\};$

$\mathcal{A}_5=\{2z^4-3cz^2+z+c^2, y-z, cx-2z^3+2cz-1\}.$

由此容易得到系统的一个覆盖. 在 MMP 中, 其覆盖可以如下得到.

```
>ps:=[z*x^2+z*y^2-c*z+1, y*x^2+(z^2-c)*y+1, (y^2+z^2-c)*x+1];
>parasolve(ps, [], [x, y, z], [c]);
```

例 6.3.5　求下列 Lorentz 系统的平衡点.

$$\begin{aligned}x_1'&=x_2(x_3-x_4)-x_1+c\\x_2'&=x_3(x_4-x_1)-x_2+c\\x_3'&=x_4(x_1-x_2)-x_3+c\\x_4'&=x_1(x_2-x_3)-x_4+c\end{aligned}$$

令

$$\mathbb{P}=\{x_2(x_3-x_4)-x_1+c, x_3(x_4-x_1)-x_2+c, x_4(x_1-x_2)-x_3+c, x_1(x_2-x_3)-x_4+c\}$$

我们有 $\mathrm{Zero}(\mathbb{P})=\cup_{i=1}^{10}\mathrm{Zero}(\mathcal{A}_i/J_i)$, 其中所有 $\mathcal{A}_i$ 为 $\mathbb{U}$ 升列. 限于篇幅, 不再具体列出. 在 MMP 中, 其覆盖可以如下得到.

```
>ps:=[x2*(x3-x4)-x1+c, x3*(x4-x1)-x2+c, x4*(x1-x2)-x3+c,
      x1*(x2-x3)-x4+c];
>parasolve(ps, [], [x4, x3, x2, x1], [c]);
```

2. 强 $\mathbb{U}$ 零点分解定理.　考虑下述例子.

例 6.3.6　令 $\mathbb{P}=\{x_1^2-u_1, (x_1-1)x_2^2-x_2+1\}$. 我们希望知道这个方程组的零点个数. 注意到 $\mathbb{P}$ 本身是一个升列, 我们有

$$\mathrm{Zero}(\mathbb{P})=\mathrm{Zero}(\mathbb{P}/(x_1-1))\cup\mathrm{Zero}(\{u_1-1, x_1-1, x_2-1\})$$

第一个分支有四个解, 第二个分支有一个解. 由此推断系统应该有五个解. 但是, 实际上这一系统一般有四个解. 如果 $u_1=1$, 第一个分支有两个解而第二分支有一个解. 此时方程有三个解. 因此, 直接由零点分解确定零点个数是不可行的.

为了给出关于参数方程第三个问题的解答, 即确定解的个数, 我们引入一种新的零点分解形式. $\mathbb{K}[\mathbb{U},\mathbb{X}]$ 中的一个升列可以写为

$$\mathcal{C}=B_1,\cdots,B_q,A_1,\cdots,A_p \tag{6.8}$$

其中 B_i 为 $\mathbb{U}$ 多项式, $A_i\in\mathbb{B}[\mathbb{X}]-\mathbb{B}$. $\mathcal{C}$ 关于 $\mathbb{X}$ 的维数定义为 $n-p$. 因此, 如果 $\mathcal{C}$ 关于 $\mathbb{X}$ 是零维的, 则 $p=n$. 升列 $\mathcal{C}$ 称为一个$\mathbb{U}$ 升列, 如果 $\mathcal{C}$ 中多项式的初式都是 $\mathbb{U}$ 多项式.

引理 6.3.7 *如果 $\mathcal{C}$ 是一个正则升列, 关于 $\mathbb{X}$ 的维数为零, 则可以构造 $\mathbb{U}$ 升列 $\mathcal{C}'$ 满足 $\operatorname{sat}(\mathcal{C})=\operatorname{sat}(\mathcal{C}')$. 进一步, 对于任何 $P\in\operatorname{sat}(\mathcal{C})$ 我们有 $\operatorname{prem}(P,\mathcal{C})=\operatorname{prem}(P,\mathcal{C}')=0$.*
令 $\mathcal{C}=B_1,\cdots,B_q,A_1,\cdots,A_n$, 其中 B_i 为 $\mathbb{U}$ 多项式且 $A_i\in\mathbb{B}[\mathbb{X}]-\mathbb{B}$. 令 $I_i=\mathrm{I}(A_i)$. 显然 $R_1=I_1$ 是一个 $\mathbb{U}$ 多项式. 记 $A'_1=A_1$. 由于 $\mathcal{C}$ 正则, $R_k=\operatorname{Res}(I_k,A_1,\cdots,A_{k-1})$ 对于 $k>1$ 是 $\mathbb{U}$ 多项式. 由结式的性质, 存在 $C_i\in\mathbb{K}[\mathbb{U},\mathbb{X}], i=1,\cdots,k$ 满足 $R_k=C_kI_k+\sum_{i=1}^{k-1}C_iA_i$. 令 $A'_k=C_kA_k+(\sum_{i=1}^{k-1}C_iA_i)x_k^{\operatorname{ldeg}(A_k)}$. 则 A'_k 的初式为 R_k. 令 $\mathcal{C}'=B_1,\cdots,B_q,A'_1,A'_2,\cdots,A'_n$. 我们可以证明 $\mathcal{C}'$ 满足引理的结论.

如果对于 $1\leqslant i\leqslant p$, $\operatorname{Res}(\mathrm{I}(A_i)\mathrm{S}(A_i),\mathcal{C})\neq 0$, 则称升列 $\mathcal{C}=A_1,\cdots,A_p$ 为*饱和的*或*具有可逆初式和隔离子*. 升列 $\mathcal{C}$ 称为关于多项式 D 饱和的, 如果 $\operatorname{prem}(D,\mathcal{C})=0$ 或 $\operatorname{Res}(D,\mathcal{C})\neq 0$.

对多项式集合 $\mathbb{P}$, 多项式 D 和关于参数的拟代数簇 $\mathbf{D}\subset\mathbb{E}^m$, 如果 $\operatorname{Zero}(\mathbb{P}/D)$ 中的零点个数对于所有可能的 $a\in\mathbf{D}$ 均是一个固定数, 我们称 $\operatorname{Zero}(\mathbb{P}/D)$ 在 $\mathbf{D}$ 上是正规的, 并利用符号 $\mathbf{N}(\operatorname{Zero}(\mathbb{P}/D))$ 表示在参数值 a 上解的个数 $|\operatorname{Zero}(\mathbb{P}/D)|$.

记 $\mathcal{C}=B_1,\cdots,B_q,A_1,\cdots,A_n$ 是关于 $\mathbb{X}$ 零维的饱和升列, 其中 B_i 为 $\mathbb{U}$ 多项式且 $\mathcal{A}=\{A_1,\cdots,A_n\}\subset\mathbb{B}[\mathbb{X}]-\mathbb{B}$. 我们假设 $I=\prod_{i=1}^n\operatorname{Res}(\mathrm{I}(A_i),\mathcal{C})\neq 0$ 和 $R=\prod_{i=1}^n\operatorname{Res}(\mathrm{S}(A_i),\mathcal{C})\neq 0$ 为 $\mathbb{U}$ 多项式. 称 $IR\neq 0$ 为 $\mathcal{C}$ 的*ID 乘积*. 显然 $\operatorname{Zero}(\mathcal{A}/IR)$ 在 $\mathbf{D}=\operatorname{Zero}(B_1,\cdots,B_q/IR)$ 和 $\mathbf{N}(\operatorname{Zero}(\mathcal{A}/IR))=\deg(\mathcal{A})=\prod_{i=1}^n\operatorname{ldeg}(A_i)$ 上是正规的.

整序原理和零点分解定理可以修改如下.

整序原理. 设 $\mathcal{C}$ 是 $\mathbb{P}$ 的特征集. 如果 $\mathcal{C}$ 是饱和的且关于 $\mathbb{X}$ 是零维的, 我们有

$$\operatorname{Zero}(\mathbb{P})=\operatorname{Zero}(\mathcal{C}/J)\cup\operatorname{Zero}(\mathbb{P}\cup\{J\})$$

其中 J 是 $\mathcal{C}$ 的 ID 乘积. 利用上述整序原理, 吴零点分解定理可以推广为下列形式.

定理 6.3.8 *对于多项式集合 $\mathbb{P}$ 和多项式 D, 我们可以找到饱和的升列 $\mathcal{C}_i$ 和多项式 D_i 满足*

$$\operatorname{Zero}(\mathbb{P}/D)=\cup_{i=1}^l\operatorname{Zero}(\mathcal{C}_i/DD_iJ_i)$$

如果 $\mathcal{C}_i$ 为正则且关于 $\mathbb{X}$ 为零维的, 则 J_i 是 $\mathcal{C}_i$ 的 ID 乘积, 否则 J_i 是 $\mathcal{C}_i$ 的初式乘积. 并且 $\mathrm{Zero}(\mathcal{C}_i/DD_iJ_i)$ 互不相交.

引理 6.3.9 设 D 为一个多项式, $\mathcal{C}$ 是一个关于 $\mathbb{X}$ 零维的饱和 $\mathbb{U}$ 升列. 假设 $\mathcal{C}$ 关于 D 是饱和的. 则我们可以构造在 $\mathbb{X}$ 上零维饱和的 $\mathbb{U}$ 升列 $\mathcal{C}_i, i=1,\cdots,r$ 和 $\mathbb{U}$ 多项式 D_i 满足

$$\mathrm{Zero}(\mathcal{C}/\mathbf{I}_{\mathcal{C}}D)=\cup\mathrm{Zero}(\mathcal{C}_i/\mathbf{I}_{\mathcal{C}_i}D_i)$$

且 $\mathrm{Zero}(\mathcal{C}_i/\mathbf{I}_{\mathcal{C}_i}D_i)$ 互不相交, 其中 $\mathbf{I}_{\mathcal{C}_i}$ 是 $\mathcal{C}_i$ 的初式乘积.

如果 D 不包含 $\mathbb{X}$ 中的变量, 上述定理显然成立. 假设 $\mathrm{prem}(D,\mathcal{C})\neq 0$. 否则 $\mathrm{Zero}(\mathcal{C}/JD)=\emptyset$. 由于 $\mathcal{C}$ 关于 D 是饱和的, $R=\mathrm{Res}(D,\mathcal{C})\neq 0$ 为 $\mathbb{U}$ 多项式. 则有

$$\begin{aligned}\mathrm{Zero}(\mathcal{C}/\mathbf{I}_{\mathcal{C}}D)&=\mathrm{Zero}(\{\mathcal{C},R\}/\mathbf{I}_{\mathcal{C}}D)\cup\mathrm{Zero}(\mathcal{C}/\mathbf{I}_{\mathcal{C}}DR)\\&=\mathrm{Zero}(\{\mathcal{C},R\}/\mathbf{I}_{\mathcal{C}}D)\cup\mathrm{Zero}(\mathcal{C}/\mathbf{I}_{\mathcal{C}}R)\end{aligned}$$

注意到上式中的两个部分互不相交. 关于 $\{\mathcal{C},R\}$, 利用定理 6.3.8, 我们有

$$\mathrm{Zero}(\{\mathcal{C},R\}/\mathbf{I}_{\mathcal{C}})=\cup_j\mathrm{Zero}(\mathcal{C}'_j/E_i\mathbf{I}_{\mathcal{C}}\mathbf{I}_{\mathcal{C}'_j})$$

其中 E_i 为多项式. 由于 $R\neq 0$, $\mathcal{C}$ 必定可约. 因此, $\mathcal{C}'_j$ 的秩低于 $\mathcal{C}$. 对

$$\mathrm{Zero}(\mathcal{C}'_j/\mathbf{I}_{\mathcal{C}}D\mathbf{I}_{\mathcal{C}_j})$$

重复上述步骤. 由引理 6.3.9, 上述过程在有限步后将停止, 所得分支互不相交.

定理 6.3.10 (强 $\mathbb{U}$ 零点分解定理) 对于多项式集合 $\mathbb{P}$ 与多项式 D, 可以找到饱和升列 $\mathcal{C}_i$ 和 $\mathbb{U}$ 多项式 D_i 满足

$$\mathrm{Zero}(\mathbb{P}/D)=\cup_{i=1}^{l}\mathrm{Zero}(\mathcal{C}_i/D_iJ_i)\bigcup\cup_{i=l+1}^{r}\mathrm{Zero}(\mathcal{C}_i/DJ_i)$$

对于 $i=1,\cdots,l$, $\mathcal{C}_i$ 为 $\mathbb{U}$ 升列且关于 $\mathbb{X}$ 为零维, J_i 是 $\mathcal{C}_i$ 的 ID 乘积. 对于 $i=l+1,\cdots,r$, $\mathcal{C}_i$ 关于 $\mathbb{X}$ 维数为正且 J_i 是 $\mathcal{C}_i$ 的初式乘积. 同时, 成员 $\mathrm{Zero}(\mathcal{C}_i/D_iJ_i), i=1,\cdots,l$ 互不相交.

3. 解的完备分类. 参数方程组解的完备分类指的是对参数空间 $\mathbb{E}^m$ 的互不相交分割, 使得当参数在一个分区上取值时, 参数方程组有相同的个数的解.

参数方程组 $\mathrm{Zero}(\mathbb{P}/D)$ 的 *精细覆盖* 是解函数的集合

$$(\mathcal{B}_i,\{\mathcal{C}_{i,j},D_{i,j}\}_{j=1,\cdots,k_i}),i=1,\cdots,d$$

满足

- $\mathcal{B}_i, i=1,\cdots,d$ 为 $\mathbb{E}^m$ 中不相交的拟代数簇, 且 $\mathcal{C}_{i,j}$ 为 $\mathbb{K}[\mathbb{U},\mathbb{X}]-\mathbb{K}[\mathbb{U}]$ 中的饱和升列, $D_{i,j}$ 为 $\mathbb{U}$ 多项式.
- $\mathrm{Proj}_{x_1,\cdots,x_n}\mathrm{Zero}(\mathbb{P}/D)=\cup_j\mathcal{B}_i$.
- 对于每一个 $u'\in\mathcal{B}_i$,

$$\mathrm{Zero}(\mathbb{P}'/D')=\cup_j\mathrm{Zero}(\mathcal{C}'_{i,j}/\mathbf{I}_{\mathcal{C}_{i,j}}*D'_{i,j})$$

  其中 $\mathbb{P}',D',\mathcal{C}'_{i,j},D'_{i,j}$ 是将 $\mathbb{U}$ 替换为 u' 所得到的结果.
- 如果 $\mathrm{Zero}(\mathbb{P}/D)$ 关于 $\mathbb{X}$ 是零维的, 则 $\mathrm{Zero}(\mathbb{P}/D)$ 在 $\mathcal{B}_i$ 上是正规的,

$$\mathbf{N}(\mathrm{Zero}(\mathbb{P}/D))=\sum_{j=1}^{k_i}\deg(\mathcal{C}_{i,j})$$

  由于 $\mathcal{B}_i$ 互不相交, $\mathrm{Zero}(\mathbb{P}/D)$ 的解的个数为 $\sum_{j=1}^{k_i}\deg(\mathcal{C}_{i,j}), i=1,\cdots,d$.

  我们可以如下构造参数方程组 $\mathrm{Zero}(\mathbb{P}/D)$ 的精细覆盖.
- 由定理 6.3.10, 我们有如下形式的分解：

$$\mathrm{Zero}(\mathbb{P}/D)=\cup_{i=1}^{l}\mathrm{Zero}(\mathcal{C}_i/D_iJ_i)\bigcup\cup_{i=l+1}^{r}\mathrm{Zero}(\mathcal{C}_i/DJ_i)$$

  其中 $\mathcal{C}_i, i=1,\cdots,l$ 为 $\mathbb{U}$ 升列且关于 $\mathbb{X}$ 是零维的, J_i 是 $\mathcal{C}_i$ 的 ID 乘积.
- 令 $T_i=\mathrm{Proj}_{x_1,\cdots,x_n}\mathrm{Zero}(\mathcal{C}_i/D_iJ_i)$, $i=1,\cdots,l$,

$$T_i=\mathrm{Proj}_{x_1,\cdots,x_n}\mathrm{Zero}(\mathcal{C}_i/DJ_i), i=l+1,\cdots,r$$

  T_i 可以有效地计算出来. $\mathcal{A}_i=\mathcal{C}_i\cap(\mathbb{K}[\mathbb{U},\mathbb{X}]-\mathbb{K}[\mathbb{U}])$.
- 对 i 做归纳法可以从 T_i 和 $\mathcal{A}_i$ 得到精细覆盖. 首先令 $\mathcal{B}_1=T_1$, $\mathcal{C}_{1,1}=\mathcal{A}_1$, $D_{1,1}=D_1$. 记

$$(\mathcal{B}_i,\{\mathcal{C}_{i,j},D_{i,j}\}_{j=1,\cdots,k_i}),\qquad i=1,\cdots,d$$

  为 T_i 和 $\mathcal{A}_i, i=1,\cdots,d$ 中得到的解函数. 我们可以通过增加 T_{d+1} 和 $\mathcal{A}_{d+1}$ 得到新的解函数集合. 对 $\mathcal{B}_i, i=1\cdots,d$, 令 $\mathcal{B}'_i=\mathcal{B}_i\cap T_{d+1}$, $\mathcal{B}''_i=\mathcal{B}_i-\mathcal{B}'_i$, $\mathcal{B}'''_i=T_{d+1}-\mathcal{B}'_i$. 则对于 $\mathcal{B}'_i$, 方程组的解可以从 $\mathcal{C}_{d+1}$ 和 $\mathcal{C}_{i,j}, j=1,\cdots,k_i$ 得到. 对于 $\mathcal{B}''_i$, 方程组的解可以从 $\mathcal{C}_{i,j}, j=1,\cdots,k_i$ 得到. 对于 $\mathcal{B}'''_i$, 方程组的解可以从 $\mathcal{C}_{d+1}$ 得到. 将 $(\mathcal{B}_i,\{\mathcal{C}_{i,j},D_{i,j}\}_{j=1,\cdots,k_i})$ 替换为三个新的解函数：$(\mathcal{B}'_i,\{\mathcal{A}_{d+1},D_{d+1}\}\cup\{\mathcal{C}_{i,j},D_{i,j}\}_{j=1,\cdots,k_i})$, $(\mathcal{B}''_i,\{\mathcal{C}_{i,j},D_{i,j}\}_{j=1,\cdots,k_i})$, $(\mathcal{B}'''_i,\{\mathcal{A}_{d+1},D_{d+1}\})$ 以便得到新的解函数集合.

在上述算法中, 我们需要计算两个拟代数簇的交集和差集.

例 6.3.11　继续考虑例 6.3.6. 由定理 6.3.10, $\mathrm{Zero}(\mathbb{P})$ 的 $\mathbb{U}$ 零点分解为 $\mathrm{Zero}(\mathbb{P}) = C_1 \cup C_2 \cup C_3 \cup C_4 \cup C_5 \cup C_6$, 其中

$C_1 = \mathrm{Zero}(\mathcal{C}/u_1(u_1-1)(16u_1-25))$, 其中 $\mathcal{C} = x_1^2 - u_1, (u_1-1)x_2^2 - (x_1+1)(x_2-1), D_1 = 1$

$C_2 = \mathrm{Zero}(\{u_1-1, x_1-1, x_2-1\})$, $D_2 = 1$

$C_3 = \mathrm{Zero}(\{u_1-1, x_1+1, 2x_2^2+x_2-1\})$, $D_3 = 1$

$C_4 = \mathrm{Zero}(\{u_1, x_1, x_2^2+x_2-1\})$, $D_4 = 1$

$C_5 = \mathrm{Zero}(\{16u_1-25, 4x_1-5, x_2-2\})$, $D_5 = 1$

$C_6 = \mathrm{Zero}(\{16u_1-25, 4x_1+5, 9x_2^2-4x_2+4\})$, $D_6 = 1$

由这一分解我们得到 $\mathbb{P} = 0$ 的下述精细覆盖：

$S_1 = \mathrm{Zero}(/u_1(u_1-1)(16u_1-25))$

$TS_{1,1} = \{x_1^2 - u_1, (u_1-1)x_2^2 - (x_1+1)(x_2-1)\}$, $D_{1,1} = 1$

$S_2 = \mathrm{Zero}(\{u_1-1\})$

$TS_{2,1} = \{x_1-1, x_2-1\}$, $D_{2,1} = 1$

$TS_{2,2} = \{x_1+1, 2x_2^2+x_2-1\}$, $D_{2,2} = 1$

$S_3 = \mathrm{Zero}(u_1)$

$TS_{3,1} = \{x_1, x_2^2+x_2-1\}$, $D_{3,1} = 1$

$S_4 = \mathrm{Zero}(16u_1-25)$

$TS_{4,1} = \{4x_1-5, x_2-2\}$, $D_{4,1} = 1$

$TS_{4,2} = \{4x_1+5, 9x_2^2-4x_2+4\}$, $D_{4,2} = 1$

从这一精细覆盖中容易看到方程组在参数区域 S_1 上有四个解, 在参数区域 S_2, S_4 上有三个解, 在参数区域 S_3 上有两个解.

§6.4　多项式方程的数值解

1. 代数方程组的复数解.　MMP 提供了求代数方程组复数解的功能. 函数 pssolver 可以给出多项式方程或方程组在复数域上的所有近似解.

下面先来看 § 6.1 中的方程 p1, 这次用 pssolver 进行求解：

```
>p1:=3*x^5-92*x^4+1016*x^3-5074*x^2+11413*x-9282:
>pssolver([p1],[x]);
```

$[[2], [13], [5.66667], [3], [7]]$

和函数 roots 求解的结果进行比较, 不难看到, pssolver 给出了方程的五个根在复数域上的近似解, 而 roots 则给出了这五个根的精确解.

从上例可以看到，函数 pssolver 的输入格式如下：

```
>pssolver(ps, vs);
```

其中 ps 为待求解的多项式的链表, vs 为变量的链表, 函数返回值是若干组解的链表, 而每一组解又放在一个子链表中. 下面再来求解 §6.1 中的方程 p6：

```
>p6:=3*x^5-92*x^4+1016*x^3-5074*x^2+1413*x-9282:
>pssolver([p6],[x]);
```

$[[16.079+8.87469e-031I],[7.34174-7.05413I],[7.34174+7.05413I],[-0.0479017-1.36161I],[-0.0479017+1.36161I]]$

该方程用函数 roots 无法求解, 而调用 pssolver, 我们获得了该方程的五个近似的复数解.

函数 pssovler 同样可以对有多个变量的多项式方程组进行求解. 考虑下面的例子：

```
>pssolver([x-2,y^2+x-6,z^2+x+y-10],[x,y,z]);
```

$[[2,2,2.44949],[2,2,-2.44949],[2,-2,3.16228],[2,-2,-3.16228]]$

该输出结果给出了方程组 $x-2=0, y^2+x-6=0, z^2+x+y-10=0$ 的四组近似解如下：

$$
\begin{aligned}
&x=2, y=2, z=2.44949\\
&x=2, y=2, z=-2.44949\\
&x=2, y=-2, z=3.16228\\
&x=2, y=-2, z=-3.16228
\end{aligned}
$$

下面再看一个求解多变量方程组的例子.

```
>pssolver([x^2+x+5,5*y+x^2-2,z^2+4*x-y^3],[x,y,z]);
[[-0.5+2.17945I,1.3+0.43589I,2.33429-1.4117I],
[-0.5+2.17945I,1.3+0.43589I,-2.33429+1.4117I],
[-0.5-2.17945I,1.3-0.43589I,2.33429+1.4117I],
[-0.5-2.17945I,1.3-0.43589I,-2.33429-1.4117I]]
```

该输出结果给出了方程组 $x^2+x+5=0, 5*y+x^2-2=0, z^2+4*x-y^3=0$ 的四组近似解如下：

$$
\begin{aligned}
&x=-0.5+2.17945I, y=1.3+0.43589I, z=2.33429-1.4117I\\
&x=-0.5+2.17945I, y=1.3+0.43589I, z=-2.33429+1.4117I\\
&x=-0.5-2.17945I, y=1.3-0.43589I, z=2.33429+1.4117I\\
&x=-0.5-2.17945I, y=1.3-0.43589I, z=-2.33429-1.4117I
\end{aligned}
$$

2. 局部数值解. 对于多变量方程组, MMP提供了另一个方程数值求解函数GSolve. 该函数既可以针对单变量多项式方程求解, 也可以针对多变量多项式方程组求解, 不过它只能求得方程或方程组在事先给定的初始值附近的一组数值解, 而不是所有数值解.

我们仍以上面提到的方程 p1 为例, 这次是用 GSolve 求解：

```
>GSolve([p1], [x], [10]);
[7]
```

GSolve(ps, vs, inis) 使用方法和 pssolver 很相似, 只是需要增加一个参数 inis, 它是一个数的链表, 代表未知数的一组初始值. GSolve 的返回值形式与 pssolver 有所不同, 它将直接给出各个未知数所对应的数值解的链表. 本例中给定的未知数的初始值是 10, 因此 GSolve 返回该方程在 10 附近的一个数值解：$x = 7$. 我们还可以通过分别设定未知数的不同初始值, 得到该方程的其他数值解：

```
>GSolve([p1], [x], [15]);
[13]
>GSolve([p1], [x], [5]);
[5.66667]
```

在这个例子中, 由于我们事先已经知道方程的解, 所以所设定的初始值都比较接近真正的解. 事实上, 用户在使用该函数时, 初始值是任意给定的, 此时一般将得到方程离这个初始值最近的那个数值解. 例如我们在上例中选定初始值为 100, 虽然这个初始值离方程所有的数值解都很远, 但是我们仍然能够得到距离 100 最近的一个数值解, 即 $x = 13$, 如下所示：

```
>GSolve([p1], [x], [100]);
[13]
```

GSolve 同样可以求解多变量多项式方程组. GSolve 给出方程组在初始值附近的一组数值解. 下面我们来看 § 6.1 中的一个求解多项式方程组的例子：

```
>p1:=x^2+y^2-41:
>p2:=x+y-9:
>GSolve([p1, p2], [x, y], [1, 0]);
```

$[5, 4]$

GSolve 的原理如下. 考虑下面方程组：

$$f_1(x_1, \cdots, x_n) = 0, \cdots, f_m(x_1, \cdots, x_n) = 0 \tag{6.9}$$

为了求解上述方程组, 我们引进函数：

$$\sigma(\mathbb{X}) = \sum_{i=1}^{m} f_i(\mathbb{X})^2 \tag{6.10}$$

其中 $\mathbb{X}=(x_1,\cdots,x_n)$. 对于给定的初值, 我们用熟知的 BFGS 方法求 $\sigma(\mathbb{X})$ 的极小值 $\mathbb{X}_0$([68] 和 [67]). 如果 $\sigma(\mathbb{X}_0)=0$, 则 $\mathbb{X}_0$ 是原方程组的一组解.

与传统的 Newton-Raphson 方法相比较, 该方法具有以下优点：方程的个数可以与变量个数无关, 对方程的 Jacobi 矩阵无限制, 而且这一方法对于任意可微分函数 f_i 均可以求解. 下面再看一个例子.

```
>vs:=[x3,x2,y1,x1,r1,r2,r3]:
>is:=[3.0,1.0,1.5,2.0,1.0,1.2,1.0]:
>ps:=[-9*y1^2+12*x1*y1-4*x1^2+13*r1^2,-y1^2-4*x1*y1+16*y1-4*x1^2+
     32*x1+5*r1^2-64,-4*x2^2+12*r2*x2+4*r2^2,-4*x3^2-4*r3*x3+
     32*x3+4*r3^2+16*r3-64,x2^2-2*x1*x2+y1^2-2*r2*y1+x1^2-
     r1^2-2*r2*r1,x3^2-2*x1*x3+y1^2-2*r3*y1+x1^2-r1^2-2*r3*r1,
     x3^2-2*x2*x3+x2^2-4*r3*r2]:
>GSolve(ps,vs,is);
[3.18963,2.06929,1.36511,2.83139,0.434738,0.626532,0.500836]
```

上面例子说明, GSolve 可以求解较大规模的方程组. 但是 GSolve 只能用于求解所给初值附近的解, 因而是一个局部求解函数. 例如, 下面调用不能求得方程 $p=0$ 的唯一一个实数解. 见图 6.1.

```
>p:=x*(x+1)*(x+2)+1;
>GSolve([p], [x], [0]);
[-0.42265]
```

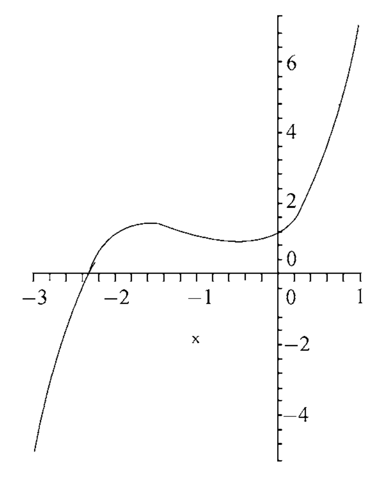


图 6.1 $y=x(x+1)(x+2)+1$ 的图像

这是因为, p^2 在 -0.42265 处有一个局部极小值. GSolve 在到达这一局部极小值后, 不能跳出这一“局部陷阱”到达其全局极值. 换一个初值, 则可以得到 $p=0$ 的解：

```
>GSolve([p], [x], [-2]);
[-2.32472]
```

3. 全局数值解. 为防止算法陷入局部最优, 我们使用全局优化方法求函数 $\sigma(X)$ 的全局极小值. 我们也可以使用下面的目标函数.

$$\sigma_a(\mathbb{X}) = \sum_{i=1}^{m} |f_i(\mathbb{X})| \tag{6.11}$$

MMP 实现的是基于遗传算法的优化方法. MMP 提供两条命令, 现介绍如下.

```
>gasolve(ps, vs, UB, LB);
>gasolve(ps, vs, UB, LB, nTime, GENERATION, POPSIZE, TOURSIZE);
```

其中

- 参数 ps 是要求解的所有方程的列表；vs 是方程中的变元列表；UB 是各变元搜索范围上界；LB 是各变元搜索范围下界.
- 第二条命令的后四个参数是遗传算法本身的参数. 各参数的含义如下. nTime：表示命令执行中遗传算法的运行次数, 一般位于 [2, 30]；GENERATION：最大循环次数；POPSIZE：群体规模；TOURSIZE：联赛选择规模, 其中要求 POPSIZ$E \geqslant$ TOURSIZE. 若用户自己指定这四个参数, 可选用第二条命令；若用户不提供这四个参数, 可选用第一条命令, 此时系统将自动选择一组默认值：nTime = 5, GENERATION = 500, POPSIZE = 10, TOURSIZE = 4.
- gasolve 的输出是一个列表 [sl, error], 其中 sl 是对应于变元 vs 的解的列表, error 是解的列表 sl 相应于方程 (6.11) 的计算结果, 即 error = $\sum_{i=1}^{m} |f_i(\mathrm{sl})|$. 本算法运行结果同时输出到当前文件夹下的“result.txt”文件.

gasolve 利用遗传算法, 求解方程组 ps 在给定搜索范围 [LB, UB] 中的数值解. 因为遗传算法是随机算法, 所以每次运行该命令所得的结果一般来说是不一样的, 单独运行一次也不一定获得满意解, 因而要多次运行. 该算法对搜索范围的大小不很敏感, 一般对一个新问题可以先给出较大的搜索范围, 再逐步缩小搜索范围对比计算结果.

有关遗传算法请参看第八章的 8.4.3.

作为局部方法 GSolve 不能很好求解的问题, 现在用全局方法来求解. 考虑前面讨论过的一个例子.

```
>p:=x*(x+1)*(x+2)+1:
>gasolve([x*(x+1)*(x+2)+1], [x], [100], [-100], 5, 100, 10, 4);
```

输出结果是 $[[-2.32472], 7.04291e-008]$. 本例用 GSolve 求解时必须选取合适的初值才能得到所需要的解.

考虑下面方程求解的例子.

$$\begin{cases} x*x/y-3=0 \\ y*y/z-4=0 \\ z*z/x-5=0 \end{cases} \tag{6.12}$$

```
>ps:=[x*x/y-3,y*y/z-4,z*z/x-5]:
>vs:=[x,y,z]:
>gasolve(ps, vs, [10, 10, 10], [-10, -10, -10], 5, 500, 10, 3);
```

变元组 vs = [x, y, z]$\in [-10, 10]$, 程序运行 5 次, 最大循环次数是 500, 群体规模是 10, 联赛规模是 3. 输出结果是 $[[3.50357, 4.09167, 4.18543], 1.69251e-009]$. 即方程的一组解是 $x = 3.50357, y = 4.09167, z = 4.18543$, 误差是 $1.69251e-009$.

考虑下面关于三角函数的方程组.

$$\begin{cases} y*sin(x)+x*3.453-12.1212=0 \\ 3*x*cos(y)-21.21*y+323.23=0 \end{cases} \tag{6.13}$$

```
>p1:=y*sin(x)+x*3.453-12.1212:
>p2:=3*x*cos(y)-21.21*y+323.23:
>gasolve([p1,p2], [x,y], [100, 100], [-100, -100], 2, 1000, 10, 3);
```

输出结果是 $[[3.03038, 14.9332], 5.20013e-009]$.

下面方程组是一个欠约束的方程组. 我们可以用 gasolve 发现其一个解.

$$\begin{cases} e^{sin(x)}+cos(z+x)-2.228=0 \\ e^{cos(y)}-sin(y+z)-1.2323=0 \end{cases} \tag{6.14}$$

```
>p1:=e^(sin(x)) + cos(z+x)-  2.228:
>p2:=e^(cos(y)) - sin(y+z) - 1.2323:
>gasolve([p1,p2], [x,y,z], [10, 10, 10], [-10, -10, -10],
         2, 500, 10, 3);
```

输出结果是 $[[-5.65906, -2.67325, -5.78556], 3.51508e-012]$.

例 6.4.1 (P4P 问题) P4P 问题即基于 4 点摄像机定位问题, 如图 6.2 所示. 令 C 为标准的摄像机中心, 并且 P_1, P_2, P_3, P_4 为参照点. 令 $c_{12} = 2\cos\angle(P_1CP_2)$, $c_{13} = 2\cos\angle(P_1CP_3)$, $c_{14} = 2\cos\angle(P_1CP_4)$, $c_{23} = 2\cos\angle(P_2CP_3)$, $c_{24} = 2\cos\angle(P_2CP_4)$, $c_{34} = 2\cos\angle(P_3CP_4)$.

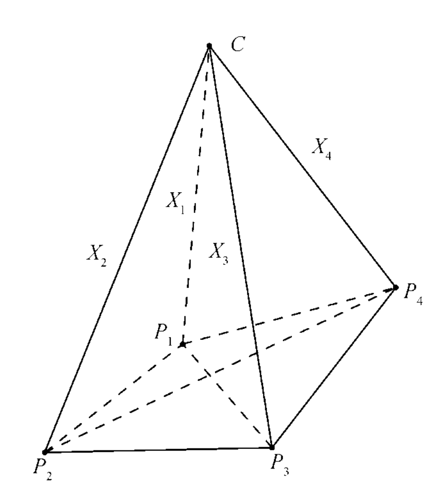


图 6.2　P4P 问题

从三角形 CP_1P_2, CP_1P_3, CP_2P_3, CP_1P_4, CP_2P_4 和 CP_3P_4, 我们得到P4P 问题的多项式系统：

$$\begin{cases} X_1^2+X_2^2-c_{12}X_1X_2-|P_1P_2|^2=0 \\ X_1^2+X_3^2-c_{13}X_1X_3-|P_1P_3|^2=0 \\ X_2^2+X_3^2-c_{23}X_2X_3-|P_2P_3|^2=0 \\ X_2^2+X_4^2-c_{24}X_2X_4-|P_2P_4|^2=0 \\ X_3^2+X_4^2-c_{34}X_3X_4-|P_3P_4|^2=0 \\ X_1^2+X_4^2-c_{14}X_1X_4-|P_1P_4|^2=0 \end{cases} \tag{6.15}$$

我们需要由方程 (6.15) 求 X_1, X_2, X_3, X_4. 这是一个过约束问题. 若摄像机的位置坐标取 (1, 1, 1), 4 个控制点的坐标分别取作 (-1, 1, 0), (-1, -1, 0), (1, -1, 0) 和 (1, 1, 0), 用 MMP 求解以上 P4P 方程系统如下：

```
>p1:=x2^2 - 2*x1^2 - 0.666667*x2 + 1.78885*x1 - 1.0:
>p2:=x3^2 - x1^2 - 0.894427*x3 + 0.894427*x1:
>p3:=x2^2 - 1.49071*x1*x2 + 0.894427*x1 - 1.0:
>p4:=-x1^2 + x3^2 - 0.4*x1*x3 + 1.78885*x1 - 2.0:
>p5:=x2^2 + x3^2 - 1.49071*x2*x3 - x1^2 + 0.894427*x1 - 1.0:
>gasolve([p1,p2,p3,p4,p5], [x1,x2,x3], [10, 10, 10], [0, 0, 0],
         5, 5000, 10, 5);
```

输出结果是 $[[2.12706, 2.85432, 2.12846], 0.00702475]$.

作为最后一个例子, 考虑图 6.3 所示的 Malfanti 问题. 相应的方程组是

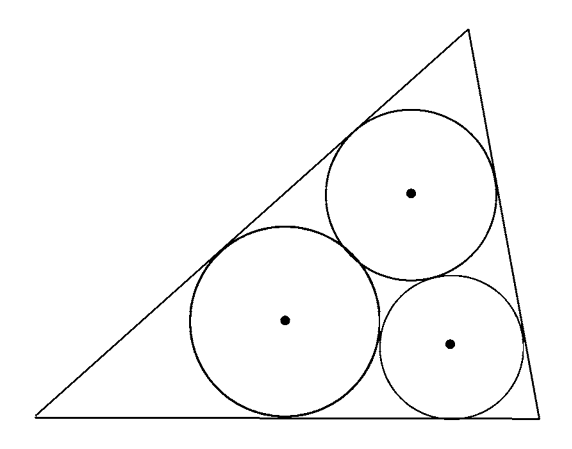


图 6.3 Malfanti 问题

$$
\left\{
\begin{aligned}
y_2^2-2y_1y_2+x_2^2-2x_1x_2+y_1^2+x_1^2-r_2^2-2r_1r_2-r_1^2=0\\
y_3^2-2y_1y_3+x_3^2-2x_1x_3+y_1^2+x_1^2-r_3^2-2r_1r_3-r_1^2=0\\
y_3^2-2y_2y_3+x_3^2-2x_2x_3+y_2^2+x_2^2-r_3^2-2r_2r_3-r_2^2=0\\
-4y_1^2+12x_1y_1-9x_1^2+13r_1^2=0\\
-y_1^2+r_1^2=0\\
-4y_3^2+12x_3y_3-9x_3^2+13r_3^2=0\\
-y_3^2+6x_3y_3-6y_3-9x_3^2+18x_3+10r_3^2-9=0\\
-y_2^2+r_2^2=0\\
-y_2^2+6x_2y_2-6y_2-9x_2^2+18x_2+10r_2^2-9=0
\end{aligned}
\right.
\tag{6.16}
$$

我们用 gasolve 对以上方程组进行求解, 在 MMP 中命令如下：

```
>p1:=y2^2-2*y1*y2+x2^2-2*x1*x2+y1^2+x1^2-r2^2-2*r1*r2-r1^2:
>p2:=y3^2-2*y1*y3+x3^2-2*x1*x3+y1^2+x1^2-r3^2-2*r1*r3-r1^2:
>p3:=y3^2-2*y2*y3+x3^2-2*x2*x3+y2^2+x2^2-r3^2-2*r2*r3-r2^2:
>p4:=-4*y1^2+12*x1*y1-9*x1^2+13*r1^2:
>p5:=-y1^2+r1^2:
>p6:=-4*y3^2+12*x3*y3-9*x3^2+13*r3^2:
>p7:=-y3^2+6*x3*y3-6*y3-9*x3^2+18*x3+10*r3^2-9:
>p8:=-y2^2+r2^2:
>p9:=-y2^2+6*x2*y2-6*y2-9*x2^2+18*x2+10*r2^2-9:
>gasolve([p1,p2,p3,p4,p5,p6,p7,p8,p9],
 [y3, x3, y2, x2, y1, x1, r3, r2, r1],
```

```
[10, 10, 10, 10, 10, 10, 10, 10, 10],
[0, 0, 0, 0, 0, 0, 0, 0, 0], 5, 5000, 10, 4);
```

输出结果是

$[[6.66134e-015, 0.528081, 8.88178e-015, 0.889796, 2.07612e-013,$
$0.364435, 0.43939, 0.104549, 0.303228], 0.976067]$

§6.5　代数方程求解的应用

6.5.1　串连机器人的逆解

1. 串连机器人的逆解.　一个 n串连机器人是通过关节将 n 个臂连结起来的开连杆. 两个臂之间的关节可以是棱形或转动关节, 臂可以沿着或围绕这个关节运动. 机器人的手是指开连杆的终端. 6R 机器人是有 6 个转动轴的机器人. 其他类似. 例如, $RRRRPR$ 机器人的前 4 个关节是转动关节, 然后有一个棱形关节, 最后是一个转动关节. 下面我们只考虑 6R 串连机器人.

如图 6.4 所示, 假设第 $i-1, i, i+1$ 三个相继关节的轴互不相交. 用 J_{i-1}, J_i, J_{i+1} 分别记这三个关节轴. 引入下面记号：

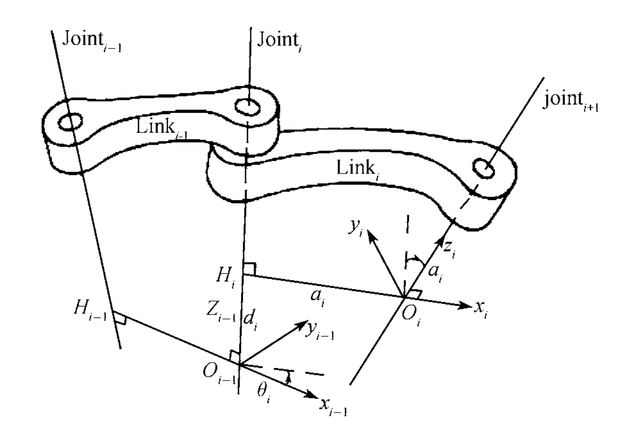


图 6.4　机器人学中逆运动方程的求解

H_iO_i, 代表 J_i 与 J_{i+1} 的公垂线, 其中 H_i 在 J_i 上, O_i 在 J_{i+1} 上. 定向是从 H_i 到 O_i. $H_{i-1}O_{i-1}$ 的含义类同.

a_i, 代表 H_iO_i 的从 H_i 到 O_i 的有向长度.

d_i, 代表沿有向轴 J_i 从 O_{i-1} 到 H_i 的有向距离.

α_i, 代表有向轴 J_i 在 O_i 点的有向平行线和有向轴 J_{i+1} 在 O_i 点的有向平行线构成的有向角.

θ_i, 代表过 O_{i-1} 点平行于 $H_{i-1}O_{i-1}$ 的直线和过 O_{i-1} 点平行于 H_iO_i 的直线构成的有向角.

d_i, a_i, α_i 称为机器人的几何约束, 它们是内在参数. θ_i 称为机器人的控制参数.

用 $F_0 = OX_0Y_0Z_0$ 代表一个附在机器人上的固定坐标系, 其中 $J_0 = J_1$ 为有向轴 OZ_0. 用 $F_i = OX_iY_iZ_i (0 < i \leqslant 6)$ 代表第 i 个坐标系, 其中 O_iX_i 沿着由 H_i 到 O_i 的方向, O_iZ_i 沿着有向轴 J_{i+1} 的方向, 而 F_i 的定向类似于 F_0 的定向. 标架 F_6 附在机器人手上. 用 $A_{i,j}$ 代表第 j 个标架 F_j 关于第 i 个标架 F_i 的矩阵. nR 机器人的 Denavit-Hartenberg 坐标系是指坐标系 $OX_iY_iZ_i = F_i, i = 1, \cdots, n$.

用一个 4×4 矩阵表示一个特定的变换, 其最后一行是 $(0,0,0,1)$, 左上角的 3×3 矩阵是旋转部分, 最后一列, 除去最下面的元素 1 后, 是平移向量. 我们有

定理 6.5.1

$$A_{i-1,i} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i * \cos\alpha_i & \sin\theta_i * \sin\alpha_i & a_i * \cos\alpha_i \\ \sin\theta_i & \cos\theta_i * \cos\alpha_i & -\cos\theta_i * \sin\alpha_i & a_i * \sin\alpha_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_{i,i-1} = \begin{bmatrix} \cos\theta_i & \sin\theta_i & 0 & -a_i \\ -\sin\theta_i * \cos\alpha_i & \cos\theta_i * \cos\alpha_i & \sin\alpha_i & -d_i * \sin\alpha_i \\ \sin\theta_i * \sin\alpha_i & -\cos\theta_i * \sin\alpha_i & \cos\alpha_i & -d_i * \cos\alpha_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

现在可以给出 6R 串连机器人的 *逆运动问题*:

问题 R 假设机器人手的位置已知, 求控制参数 θ_i, 使得机器人的手恰在已知的位置.

假设已知手位置由矩阵

$$H = A_{0,6} = \begin{bmatrix} h_{11} & h_{21} & h_{31} & h_{41} \\ h_{12} & h_{22} & h_{32} & h_{42} \\ h_{13} & h_{23} & h_{33} & h_{43} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

给出, 则问题等价于解关于 θ_i 的下列矩阵方程

$$A_{01} \circ A_{12} \circ \cdots \circ A_{56} = H \tag{6.17}$$

其中 h_{ij} 已知.

2. 一种特殊串连机器人的逆解.　一般的 6R 机器人逆运动方程的求解是非常复杂的. 我们仅讨论几种简化类型的机器人. 一个 6R 机器人是 Puma型的当且仅当其 6 个关节轴相继对或是相互平行, 或是相互垂直. 总共有 2^5 个不同的 Puma 型机器人. 为明确起见, 我们考虑满足下面条件的 Puma 型机器人：

$$J_1 \perp J_2 \parallel J_3 \perp J_4 \perp J_5 \perp J_6$$

参看图 6.5.

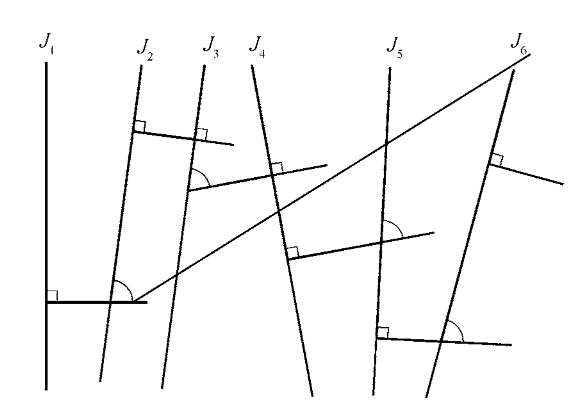


图 6.5　Puma 型机器人

对于 Puma 型机器人, 我们首先给出关节轴和 O_iH_i 定向, 使得所有的直角 α_i 均为 $+\dfrac{\pi}{2}$. 这样相继的约束角将是

$$(\alpha_1, \alpha_2, \cdots, \alpha_6) = \left(+\frac{\pi}{2}, 0, +\frac{\pi}{2}, +\frac{\pi}{2}, +\frac{\pi}{2}, 0\right)$$

此外, 我们还将选择标架 F_0 和 F_6, 使得 O_0Z_0 位于有向轴 J_1 上, 而且与之同向, $O_6 = H_6$ 与 O_5 重合, O_6Z_6 与 O_5Z_5 具有相同的方向. 由此引出简化方程：$d_6 = 0$. 现在令

$$\begin{cases} c_i = \cos\theta_i, & s_i = \sin\theta_i, & i = 1, 3, 4, 5, 6 \\ c_{23} = \cos\theta_{23}, & s_{23} = \sin\theta_{23} \\ \theta_{23} = \theta_2 + \theta_3, & d_{23} = d_2 + d_3 \end{cases}$$

则求解 Puma 型机器人的问题等价于利用 a_i, d_i 求解 $c_i, s_i (i \neq 2)$ 和 c_{23}, s_{23}, 而 h_{ij} 看作是已知参数. 为此, 我们用一个与 (6.17) 等价的方程

$$A_{34} \circ A_{45} \circ A_{56} = A_{32} \circ A_{21} \circ A_{10} \circ H \tag{6.18}$$

代替方程 (6.17). 为简单起见, 引进多项式:

$$\begin{cases} f_{i1} = s_1 * h_{i2} + c_1 * h_{i1} - \delta_{4i} * a_1, & i = 1,2,3,4 \\ f_{i2} = -s_1 * h_{i1} + c_1 * h_{i2} + \delta_{4i} * d_{23}, & i = 1,2,3,4 \\ f_{43} = h_{43} - d_1 \end{cases} \tag{6.19}$$

由此得到多项式组：$P_i = 0, i = 1, \cdots, 12$, 其中

$$\begin{cases} P_1 = s_6 * s_4 + c_6 * c_5 - s_{23} * h_{13} - c_{23} * f_{11} \\ P_2 = s_6 * c_4 - c_6 * c_5 * s_4 - f_{12} \\ P_3 = c_6 * s_5 - s_{23} * f_{11} + c_{23} * h_{13} \\ P_4 = s_6 * c_5 * c_4 - c_6 * s_4 + s_{23} * h_{23} + c_{23} * f_{21} \\ P_5 = s_6 * c_5 * s_4 + c_6 * c_4 - f_{22} \\ P_6 = s_6 * s_5 + s_{23} * f_{21} - c_{23} * h_{23} \\ P_7 = s_5 * c_4 - s_{23} * h_{33} - c_{23} * f_{31} \\ P_8 = s_5 * s_4 + f_{32} \\ P_9 = c_5 + s_{23} * f_{31} - c_{23} * h_{33} \\ P_{10} = c_5 * c_4 * a_5 + s_4 * d_5 + c_4 * a_4 + c_3 * a_2 - s_{23} * f_{43} \\ \qquad -c_{23} * f_{41} + a_3 \\ P_{11} = c_5 * s_4 * a_5 + s_4 * a_4 - c_4 * d_5 + f_{42} \\ P_{12} = s_5 * a_5 + s_3 * a_2 - s_{23} * f_{41} + c_{23} * f_{43} + d_4 \end{cases} \tag{6.20}$$

除此之外, 对于三角函数我们还有三角恒等式：$P_i = 0, i = 13, \cdots, 18$, 其中

$$\begin{cases} P_i = c_{i-12}^2 + s_{i-12}^2 - 1, & i = 13, 15, 16, 17, 18 \\ P_{14} = c_{23}^2 + s_{23}^2 - 1 \end{cases}$$

于是, Puma 型机器人的逆问题等价于确定多项式集合

$$\mathbb{P} = \{P_1, \cdots, P_{18}\} \tag{6.21}$$

的零点集 $\mathrm{Zero}(\mathbb{P})$.

为了确定零点集 $\mathrm{Zero}(\mathbb{P})$, 我们先做变换化简原问题. 借助于方程 $P_{16} = 0$, 多项式 P_1 和 P_2 分别等价于多项式 P_1' 和 P_2', 其中

$$P_1' = s_4 * P_1 + c_4 * P_2, \quad P_2' = s_4 * P_1 - c_4 * P_2$$

类似地, 多项式 P_4, P_5 和 P_7, P_8 分别等价于多项式 P_4', P_5' 和 P_7', P_8', 其中

$$\begin{aligned} P_4' &= -s_4 * P_4 + c_4 * P_5, \quad & P_5' &= c_4 * P_4 + s_4 * P_5 \\ P_7' &= s_4 * P_7 - c_4 * P_8, \quad & P_8' &= c_4 * P_7 + s_4 * P_8 \end{aligned}$$

进一步地, 多项式 P_2', P_3 和 P_5', P_6 分别等价于多项式 P_2'', P_3' 与 P_5'', P_6', 其中

$$P_2'' = s_5 * P_2' - c_5 * P_3, \quad P_3' = c_5 * P_2' + s_5 * P_3$$
$$P_5'' = s_5 * P_5' - c_5 * P_6, \quad P_6' = c_5 * P_5' + s_5 * P_6$$

将 P_i 写成 $P_i', i \geqslant 9$, 重写 P_2'', P_5'' 成 P_2', P_5', 则多项式集 $\mathbb{P}$ 等价于多项式集

$$\mathbb{P}' = \{P_1', \cdots, P_{18}'\} \tag{6.22}$$

其中

$$\begin{aligned}
P_1' &= s_6 - c_4 * f_{12} - s_4 * c_{23} * f_{11} - s_4 * s_{23} * h_{13} \\
P_2' &= c_6 - c_5 * c_4 * (c_{23} * f_{11} - s_{23} * h_{13}) \\
&\quad + c_5 * s_4 * f_{12} + s_5 * c_{23} * h_{13} - s_5 * s_{23} * f_{11} \\
P_3' &= -c_5 * c_{23} * h_{13} + c_5 * s_{23} * f_{11} - s_5 * c_4 * c_{23} * f_{11} \\
&\quad - s_5 * c_4 * s_{23} * h_{13} + s_5 * s_4 * f_{12} \\
P_4' &= c_6 - c_4 * f_{22} - s_4 * c_{23} * f_{21} - s_4 * s_{23} * h_{23} \\
P_5' &= s_6 + c_5 * c_4 * c_{23} * f_{21} + c_5 * c_4 - s_5 * s_{23} * h_{23} \\
&\quad - c_5 * s_4 * f_{22} - s_5 * c_{23} * h_{23} + s_5 * s_{23} * f_{21} \\
P_6' &= c_5 * c_{23} * h_{23} - c_5 * s_{23} * f_{21} + s_5 * c_4 * c_{23} * f_{21} \\
&\quad + s_5 * c_4 * s_{23} * h_{23} - s_5 * s_4 * f_{22} \\
P_7' &= c_4 * f_{32} + s_4 * c_{23} * f_{31} + s_4 * s_{23} * h_{33} \\
P_8' &= s_5 - c_4 * c_{23} * f_{31} - c_4 * s_{23} * h_{33} + s_4 * f_{32} \\
P_i' &= P_i, i = 9, \cdots, 18
\end{aligned}$$

这样, 求 $\mathrm{Zero}(\mathbb{P})$ 等价于求 $\mathrm{Zero}(\mathbb{P}')$.

考虑更为特殊的 Puma560型机器人. Puma560 是一个 6R 机器人, 其关节轴 J_i 还满足下列关系：

$$J_4, J_5, J_6 \text{平行而且任何两个都不重合}$$

上述条件以及机器人手所处的自然几何形态蕴涵下面约束：

$$a_4 = a_5 = d_5 = 0$$

另一方面, 我们将假定：a_2, a_3, d_{23}, d_4 均不为零. 于是, 多项式 $P_{10}', P_{11}', P_{12}'$ 将被简化为下列形式：

$$\begin{aligned}
P_{10}' &= c_3 * a_2 - s_{23} * f_{43} - c_{23} * f_{41} + a_3 \\
P_{11}' &= s_1 * h_{41} - c_1 * h_{42} - d_{23} \\
P_{12}' &= s_3 * a_2 - s_{23} * f_{41} + c_{23} * f_{43} + d_4
\end{aligned} \tag{6.23}$$

现在, 我们引进诸变量 c, s 的序如下：$s_1 < c_1 < s_{23} < c_{23} < s_4 < c_4 < s_5 < c_5 < s_6 < c_6 < s_3 < c_3$, 则用特征集算法很容易给出其三角型特征列：

$$CS = \{C_1, \cdots, C_{12}\}$$

其中 C_i 给出如下：

$$\begin{cases} C_1 = s_1^2 * (h_{42}^2 + h_{41}^2) - 2 * s_1 * h_{41} d_{23} - h_{42}^2 + d_{23}^2 \\ C_2 = c_1 * h_{42} - s_1 * h_{41} + d_{23} \\ C_3 = 4 * s_{23}^2 * (f_{43}^2 + f_{41}^2) * (d_4^2 + a_3^2) \\ \qquad -4 * s_{23} * (f_{43} * a_3 + f_{41} * d_4) * (f_{23}^2 + f_{41}^2 + d_4^2 + a_3^2 - a_2^2) \\ \qquad +(f_{23}^2 + f_{41}^2 + d_4^2 + a_3^2 - a_2^2)^2 - 4 * (f_{43} * d_4 - f_{41} * a_3)^2 \\ C_4 = 2 * c_{23} * (f_{43} * d_4 - f_{41} * a_3) - 2 * s_{23} * (f_{43} * a_3 + f_{41} * d_4) \\ \qquad +f_{43}^2 + f_{41}^2 + d_4^2 + a_3^2 - a_2^2 \\ C_5 = s_4^2 * (c_{23} * f_{31} + s_{23} * h_{33})^2 + s_4^2 * f_{23}^2 - f_{32}^2 \\ C_6 = c_4 * f_{32} + s_4 * c_{23} * f_{31} + s_4 * s_{23} * h_{33} \\ C_7 = s_5 - c_4 * c_{23} * f_{31} - c_4 * s_{23} * h_{33} + s_4 * f_{32} \\ C_8 = c_5 - c_{23} * h_{33} + s_{23} * f_{31} \\ C_9 = s_6 - c_4 * f_{12} - s_4 * c_{23} * f_{11} - s_4 * s_{23} * h_{13} \\ C_{10} = c_6 - c_4 * f_{22} - s_4 * c_{23} * f_{21} - s_4 * s_{23} * h_{23} \\ C_{11} = s_3 * a_2 + c_{23} * f_{43} - s_{23} * f_{41} + d_4 \\ C_{12} = c_3 * a_2 - c_{23} * f_{41} - s_{23} * f_{43} + a_3 \end{cases} \tag{6.24}$$

在 MMP 中, 以上特征列可以通过下面的指令得到：

```
>P_1:=s_6-c_4*f_12-s_4*c_23*f_11-s_4*s_23*h_13;
>P_2:=c_6-c_5*c_4*(c_23*f_11-s_23*h_13)+c_5*s_4*f_12
      +s_5*c_23*h_13-s_5*s_23*f_11;
>P_3:=-c_5*c_23*h_13+c_5*s_23*f_11-s_5*c_4*c_23*f_11
      -s_5*c_4*s_23*h_13+s_5*s_4*f_12;
>P_4:=c_6-c_4*f_22-s_4*c_23*f_21-s_4*s_23*h_23;
>P_5:=s_6+c_5*c_4*c_23*f_21+c_5*c_4-s_5*s_23*h_23
      -c_5*s_4*f_22-s_5*c_23*h_23+s_5*s_23*f_21;
>P_6:=c_5*c_23*h_23-c_5*s_23*f_21+s_5*c_4*c_23*f_21
      +s_5*c_4*s_23*h_23-s_5*s_4*f_22;
>P_7:=c_4*f_32+s_4*c_23*f_31+s_4*s_23*h_33;
>P_8:=s_5-c_4*c_23*f_31-c_4*s_23*h_33+s_4*f_32;
```

```
>P_9:=c_5+s_23*f_31-c_23*h_33;
>P_10:=c_3*a_2-s_23*f_43-c_23*f_41+a_3;
>P_11:=s_1*h_41-c_1*h_42-d_23;
>P_12:=s_3*a_2-s_23*f_41+c_23*f_43+d_4;
>P_13:=c_1^2+s_1^2-1;
>P_14:=c_23^2+s_23^2-1;
>P_15:=c_3^2+s_3^2-1;
>P_16:=c_4^2+s_4^2-1;
>P_17:=c_5^2+s_5^2-1;
>P_18:=c_6^2+s_6^2-1;
>ps:=[P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_10, P_11, P_12,
 P_13, P_14, P_15, P_16, P_17, P_18];
>vs:=[c_3, s_3, c_6, s_6, c_5, s_5, c_4, s_4, c_23, s_23, c_1, s_1];
>charset(ps, vs, [], "weak");
```

去掉一些非零因子后, 诸 C_i 的非平凡初式为

$$\begin{aligned}
&I_1 = h_{42}^2 + h_{41}^2\\
&I_2 = h_{42}\\
&I_3 = f_{43}^2 + f_{41}^2\\
&I_4 = f_{43} * d_4 - f_{41} * a_3\\
&I_5 = (c_{23} * f_{31} + s_{23} * h_{33})^2 + f_{32}^2\\
&I_6 = f_{32}
\end{aligned} \tag{6.25}$$

我们这里只考虑实解. 所以, 由 $I_2 \neq 0$ 可推得 $I_1 \neq 0$, $I_6 \neq 0$ 意味着 $I_5 \neq 0$ 与 $I_3 \neq 0$. 于是由整序原理得到

定理 6.5.2　对于 Puma560 型机器人, 与逆运动方程 $\mathbb{P} = 0$ 相应的 (6.21) 中多项式集 $\mathbb{P}$ 的特征集 CS 可由 (6.24) 和 (6.24) 给出, 如果考虑实数解, 则有

$$\begin{aligned}
\mathrm{Zero}(\mathbb{P}) = &\mathrm{Zero}(CS/f_{32} * h_{42} * I_4)\\
&\cup \mathrm{Zero}(\mathbb{P} \cup \{h_{42}\}) \cup \mathrm{Zero}(\mathbb{P} \cup \{f_{32}\}) \cup \mathrm{Zero}(\mathbb{P} \cup \{I_4\})
\end{aligned} \tag{6.26}$$

(6.26) 中后三个零点集的求解可以类似计算, 这里将不予考虑. 从特征集可以看出, 对于给定的手位置, 几何上至多有 8 个可能的实解.

6.5.2　广义 Stewart 平台的正解

3. 广义 Stewart 平台.　Stewart平台由两个刚体组成 (图 6.6). 下面的刚体称为 基体, 上面的刚体称为 动平台, 简称为 平台. 基体和平台由六条可伸缩的操纵杆连结.

基体的位置和方向是固定的. 通过调节操纵杆的长度, 平台可以在空中移动. 六个操纵杆的长度 $l_1, l_2, l_3, l_4, l_5, l_6$ 称为 *控制参数* 或 *驱动参数*. 对于给定的一组控制参数, 平台的位置和方向一般可以被确定.

Stewart 平台是并联机械机构的代表. 串联机构具有工作空间大, 灵活度高的特点. 而并联机构具有刚度大, 承载能力大, 精度高的特点. 两种机构的优缺点是互补的. 历史上, 串联机构的使用较多, 相关的研究也较早. Stewart 平台源自 20 世纪 50 年代 Stewart 为飞行器模拟试验所设计的机械机构与 Gough 设计的汽车轮胎测试机构. 近 20 年来, Stewart 平台逐渐成为机构学研究的热点. 主要原因是, 20 世纪 80 年代, 各国相继推出基于 Stewart 平台的数控机床, 被称为 "机床领域的革命", "21 世纪的数控机床". 这种机床充分利用了 Stewart 平台刚性好, 承载能力大的优点, 希望制造出精度高而又结构简单的数控机床. Stewart 平台还被用于诸多领域：机器人, 数控机床, 纳米技术等领域, 已经成为一种使用广泛的机构.

机械与机构的运动分析主要有两个问题：(1) 正解问题：给定控制参数求末端的位置. (2) 逆解问题：给定末端的位置求控制参数.

Stewart 平台运动分析的逆解问题比较容易. 给定了平台的位置, 可以直接计算出六个控制参数, 即操纵杆的长度. 正解问题则十分困难. 具体讲 Stewart 平台的 *正解* 即由给定的基体的位置与六个驱动参数的值, 怎样唯一确定平台的位置. 关于 Stewart 平台研究的大部分工作是其正解问题的研究, 而且正解问题目前还没有彻底解决, 成为 Stewart 平台某些应用的瓶颈问题.

一个自然的想法是, 我们是否可以用其他的几何约束代替点到点的距离作为驱动参数呢? 例如, 我们可以用动平台上六个点到基体上的六个平面之间的距离来作为驱动参数. 这样就得到一种新的 Stewart 平台 (图 6.7). 当然, 我们还可以考虑其他类型的几何约束作为驱动参数. 空间中的一个刚体有六个自由度. 因此确定一个刚体在空间中的位置和定向, 我们需要六个几何约束. 由此引出下面的定义.

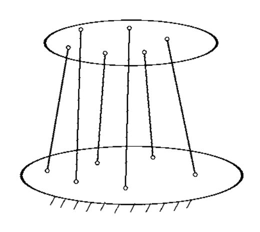




图6.6 Stewart平台

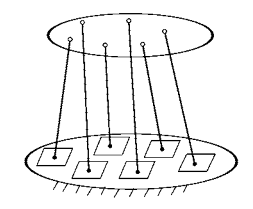


图6.7 基于点/面距离的广义Stewart平台

定义 6.5.3 *一个广义Stewart平台由两个刚体以及连接两个刚体的六个几何约束构成. 固定的刚体称为基体; 另一个随着约束的数值的变化而移动的刚体称为移动平台. 以后简称为平台.*

在 3D 情形, 我们考虑三种 几何对象：点, 线, 面; 两类 几何约束：点点, 点线, 点面, 线线间的距离约束和线线, 线面, 面面间的角度约束. 根据所用约束种类的不同类型, 可以将广义 Stewart 平台分为以下四类：

3D3A 驱动参数为三个距离约束和三个角度约束.

4D2A 驱动参数为四个距离约束和两个角度约束.

5D1A 驱动参数为五个距离约束和一个角度约束.

6D 驱动参数为六个距离约束.

空间中的一个刚体最多需要三个角度约束便可以确定它的方向. 不需考虑四个以上的角度约束. 假定基体和平台上的几何对象都是不同的, 可以证明：有 1120 种 **3D3A** 广义 Stewart 平台, 1260 种 **4D2A** 广义 Stewart 平台, 1008 种 **5D1A** 广义 Stewart 平台和 462 种 **6D** 广义 Stewart 平台. 共有 3850 种广义 Stewart 平台.

4. 3D3A 广义 Stewart 平台的角度约束. 一般广义 Stewart 平台解析解的求解是非常困难的. 对于 **3D3A** 广义 Stewart 平台, 我们则可以用吴零点分解定理求出其解析解. 主要原因是 **3D3A** 广义 Stewart 平台是可以解偶控制的. 具体讲其正解可以通过两个步骤求解. 我们可以先用三个角度约束来确定平台的三个旋转自由度. 再增加上平移约束不会破坏先前加入的旋转约束. 这样, 我们可以继续加入三个距离约束来确定平台的位置. 按照这一思路, 我们可以得到 1120 种广义 Stewart 平台 **3D3A** 正解的解析解.

首先介绍怎样求解三个角度约束. 描述三个角度约束的方程都是线性的. 角度约束的表示只与平行于直线的单位向量或垂直于平面的单位法向量有关. 这样, 我们只需考虑两个单位向量间的角度约束.

假设 L_1, L_2, L_3 是基体上的三条直线, L_4, L_5, L_6 是平台上的三条直线. 假设旋转矩阵是 $\mathbf{R} = (r_{ij})_{3\times3}$, 角度约束是

$$\cos(\angle(L_1, L_4)) = d_1, \cos(\angle(L_2, L_5)) = d_2, \cos(\angle(L_3, L_6)) = d_3$$

令 $\mathbf{s_i}$ 是平行于 $L_i (i = 1, \cdots, 6)$ 的单位向量. 我们得到下面的方程系统：

$$\begin{cases} \mathbf{R^T R} = \mathbf{I}, \det(\mathbf{R}) = 1 \\ \mathbf{s_1} \cdot \mathbf{R s_4} = d_1, \mathbf{s_2} \cdot \mathbf{R s_5} = d_2, \mathbf{s_3} \cdot \mathbf{R s_6} = d_3 \end{cases} \tag{6.27}$$

如果约束中的几何对象是平面而不是直线, 我们得到与上面一样的方程系统. 所以可以将上面的方程系统作为基体和平台之间三个角度约束的一般系统. 因为 L_i 是固定在基体和平台上的, 我们可以不失一般性地假定：

$$\mathbf{s_1} = (0, 0, 1), \mathbf{s_2} = (0, m_0, n_0), \mathbf{s_3} = (l_1, m_1, n_1)$$
$$\mathbf{s_4} = (0, 0, 1)^T, \mathbf{s_5} = (0, m_2, n_2)^T, \mathbf{s_6} = (l_3, m_3, n_3)^T$$

在 MMP 中, 方程系统 (6.27) 可以通过下面的一系列指令得到.
以下运算可以得到关于 $\mathbf{R}^{\mathbf{T}}\mathbf{R}=\mathbf{I}$ 的 9 个方程：

```
>R:=matrix(3, 3, [r11, r12, r13, r21, r22, r23, r31, r32, r33]);
>P:=multiply(tran(R), R);
>h1:=getval(P, 1, 1)-1;
>h2:=getval(P, 1, 2);
>h3:=getval(P, 1, 3);
>h4:=getval(P, 2, 1);
>h5:=getval(P, 2, 2)-1;
>h6:=getval(P, 2, 3);
>h7:=getval(P, 3, 1);
>h8:=getval(P, 3, 2);
>h9:=getval(P, 3, 3)-1;
```

以下运算可以得到方程 $\det(\mathbf{R})=1$:

```
>h10:=det(R)-1;
```

关于 3 个角度约束的 3 个方程可以如下得到：

```
>h11:=expand(getval(multiply(matrix([0, 0, 1], 1),
multiply(R, matrix([0, 0, 1], 0))), 1, 1)-d1);
>h12:=expand(getval(multiply(matrix([0, m0, n0], 1),
multiply(R, matrix([0, m2, n2], 0))), 1, 1)-d2);
>h13:=expand(getval(multiply(matrix([l1, m1, n1], 1),
multiply(R, matrix([l3, m3, n3], 0))), 1, 1)-d3);
```

方程 $h_i=0, i=1,\cdots,13$ 可以利用吴特征列方法约化成三角列形式. 在 MMP 中命令如下：

```
>ps:=[h1, h2, h3, h4, h5, h6, h7, h8, h9, h10, h11, h12, h13];
>vs:=[r33, r11, r12, r13, r21, r22, r23, r31, r32];
>wsolve(ps, vs, [], "weak");
```

约化的结果如下：

$$
\begin{aligned}
&h_1=\sum\nolimits_{i=0}^{8} z_{1i}\mathbf{r}_{32}{}^{i}\\
&h_2=z_2\mathbf{r}_{31}+z_{24}r_{32}^3+z_{25}r_{32}^2+z_{26}r_{32}+z_{27}=0\\
&h_3=z_{30}\mathbf{r}_{23}+z_{31}(z_{115}r_{32}^3+z_{116}r_{32}^2+z_{117}r_{32}+z_{118})r_{31}^2\\
&\qquad+(z_{119}r_{32}^4+z_{120}r_{32}^3+z_{121}r_{32}^2+z_{122}r_{32}+z_{123})r_{31}+z_{124}r_{32}^5+z_{125}r_{32}^4\\
&\qquad+z_{126}r_{32}^3+z_{127}r_{32}^2+z_{128}r_{32}+z_{129}=0\\
&h_4=m_0m_2\mathbf{r}_{22}+q_2r_{23}+q_3r_{32}+d_1n_0n_2-d_2=0
\end{aligned}
$$

$$
\begin{aligned}
h_5 &= z_5\mathbf{r}_{21} - d_1 r_{22} r_{23} r_{31} - r_{31} r_{32} + r_{23}^2 r_{31} r_{32} = 0 \\
h_6 &= z_6\mathbf{r}_{13} + r_{31} r_{23}^2 + (d_1^2 - 1) r_{31} = 0 \\
h_7 &= r_{31}\mathbf{r}_{12} - d_1 r_{32} r_{22} + (r_{32}^2 - 1) r_{23} = 0 \\
h_8 &= \mathbf{r}_{11} + r_{32} r_{23} - r_{22} d_1 = 0 \\
h_9 &= \mathbf{r}_{33} = d_1
\end{aligned}
\tag{6.28}
$$

这里 $z_2 = z_{20} r_{32}^3 + z_{21} r_{32}^2 + z_{22} r_{32} + z_{23}$, $q_2 = m_0 n_2$, $q_3 = n_0 m_2$, $z_5 = (d_1^2 - 1) r_{22} - r_{23} r_{32} d_1$, $z_6 = (d_1^2 - 1) r_{22} - r_{23} r_{32} d_1$, 所有的 $z_{i,j}$ 都是关于参数 l_k, m_s, n_t, d_l 的多项式, 具体形式省略.

定理 6.5.4 *加入三个角度约束后, 对应于平台的方向的解至多为 8. 这个界对于实解而言是最佳的.*

证明 求解方程系统 (6.27), 一般情形下我们只需求解方程 (6.28), 它具有三角形式. 我们可以从 $h_1 = 0, h_2 = 0, \cdots, h_9 = 0$ 中依次求出变量 $r_{32}, r_{31}, r_{23}, r_{22}, r_{21}, r_{13}, r_{12}, r_{11}, r_{33}$ 的值. 因为 $h_i, i = 2, \cdots, 9$ 关于他们的主变量是线性的, h_1 关于其主变量 r_{32} 是八次的, 上面的方程系统至多有八个解. 根据下面的例 6.5.5, 这个问题确实可以有八个实解. 因此, 在一般情形这个上界无法再改进.

例 6.5.5 *图 6.8 表示一个* **3D3A** *广义 Stewart平台. L_1, L_2, L_3 是平台上的直线, L_4, L_5, L_6 是基体上的直线. a 和 d 分别表示角度和距离约束. 进一步假定 L_1, L_2 和 L_3 彼此垂直, L_4, L_5 和 L_6 也彼此垂直.*

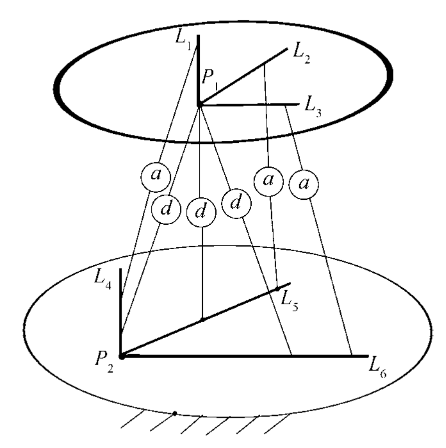


图 6.8 一个 **3D3A** 广义 Stewart 平台

角度约束是 $\cos(\angle(L_1, L_4)) = d_1$, $\cos(\angle(L_2, L_5)) = d_2$, $\cos(\angle(L_3, L_6)) = d_3$. *如果选取* $\mathbf{s_1} = (0,0,1)$, $\mathbf{s_2} = (0,1,0)$, $\mathbf{s_3} = (1,0,0)$, $\mathbf{s_4} = (0,0,1)^T$, $\mathbf{s_5} = (0,1,0)^T$, $\mathbf{s_6} = (1,0,0)^T$, *根据方程系统 (6.27), 我们得到* $r_{11} = d_3$, $r_{22} = d_2$, $r_{33} = d_1$. *在MMP中, 本例的方程系统 (6.27) 可以如下得到:*

```
>R:=matrix(3, 3, [d3, r12, r13, r21, d2, r23, r31, r32, d1]);
>P:=multiply(tran(R), R);
>h1:=getval(P, 1, 1)-1;
>h2:=getval(P, 1, 2);
>h3:=getval(P, 1, 3);
>h4:=getval(P, 2, 1);
>h5:=getval(P, 2, 2)-1;
>h6:=getval(P, 2, 3);
>h7:=getval(P, 3, 1);
>h8:=getval(P, 3, 2);
>h9:=getval(P, 3, 3)-1;
>h10:=det(R)-1;
>h11:=expand(getval(multiply(matrix([0, 0, 1], 1),
multiply(R, matrix([0, 0, 1], 0))), 1, 1)-d1);
>h12:=expand(getval(multiply(matrix([0, 1, 0], 1),
multiply(R, matrix([0, 1, 0], 0))), 1, 1)-d2);
>h13:=expand(getval(multiply(matrix([1, 0, 0], 1),
multiply(R, matrix([1, 0, 0], 0))), 1, 1)-d3);
```

方程系统 $h_i, i = 1, \cdots, 13$ 可以利用吴特征列方法以变量序 $r_{32} < r_{31} < r_{23} < r_{21} < r_{13} < r_{12}$ 化为三角列形式. 在MMP中命令如下:

```
>ps:=[h1, h2, h3, h4, h5, h6, h7, h8, h9, h10];
>vs:=[r12, r13, r21, r23, r31, r32];
>wsolve(ps, vs, [], "weak");
```

由于 $\mathrm{h11} = 0, \mathrm{h12} = 0, \mathrm{h13} = 0$, 所以 ps 中没有包含这 3 个方程. 约化结果如下:

$$\left\{\begin{array}{l}
r_{12} * d_3 + d_2 * r_{21} + r_{32} * r_{31} = 0 \\
r_{13} * d_3 + r_{23} * r_{21} + d_1 * r_{31} = 0 \\
r_{32}^2 * d_3^2 - d_3^2 + 2 * r_{32} * r_{31} * d_2 * r_{21} \\
\quad - r_{31}^2 * d_2^2 + d_2^2 + r_{32}^2 * r_{31}^2 = 0 \\
d_3 - d_1 * d_2 + r_{32} * r_{23} = 0 \\
r_{32}^2 * d_3^2 - d_3^2 + 2 * d_1 * d_2 * d_3 \\
\quad + r_{31}^2 * d_2^2 - d_2^2 + r_{32}^2 * r_{31}^2 = 0 \\
r_{32}^2 * d_3^2 - d_3^2 + 2 * d_1 * d_2 * d_3 \\
\quad - r_{32}^2 * d_2^2 - d_1^2 * d_2^2 - r_{32}^4 \\
\quad - d_1^2 * r_{32}^2 + r_{32}^2 = 0
\end{array}\right. \tag{6.29}$$

令 $\angle(L_1, L_4)) = \frac{\pi}{3}$, $\angle(L_2, L_5) = \frac{\pi}{3}$, $\angle(L_3, L_6) = \frac{\pi}{3}$. 那么 $r_{33} = d_1 = \frac{1}{2}$, $r_{22} = d_2 = \frac{1}{2}$, $r_{11} = d_3 = \frac{1}{2}$. 代入三个角度约束 d_1, d_2, d_3 的具体值, 对以上三角列进行求解. 在 *MMP* 中, 求解命令如下：

```
>d1:=1/2;
>d2:=1/2;
>d3:=1/2;
>p1:=r12*d3+d2*r21+r32*r31;
>p2:=r13*d3+r23*r21+d1*r31;
>p3:=r32^2*d3^2-d3^2+2*r32*r31*d2*r21-r31^2*d2^2+d2^2+r32^2*r31^2;
>p4:=d3-d1*d2+r32*r23;
>p5:=r32^2*d3^2-d3^2+2*d1*d2*d3+r31^2*d2^2-d2^2+r32^2*r31^2;
>p6:=r32^2*d3^2-d3^2+2*d1*d2*d3-r32^2*d2^2-d1^2*d2^2-r32^4-
    d1^2*r32^2+r32^2;
>roots([p1,p2,p3,p4,p5,p6],[r12,r13,r21,r23,r31,r32]);
```

我们得到关于旋转矩阵 R 的实解的八个集合列在表 *6.1* 中.

表 6.1 旋转矩阵 R 的八个实解

r_{12}	r_{13}	r_{21}	r_{23}	r_{31}	r_{32}
$\frac{1}{-1+\sqrt{5}}$	$\frac{-1+\sqrt{5}}{4}$	$\frac{-3+\sqrt{5}}{2(-1+\sqrt{5})}$	$\frac{1}{1-\sqrt{5}}$	$\frac{1}{1-\sqrt{5}}$	$\frac{-1+\sqrt{5}}{4}$
$\frac{1}{1-\sqrt{5}}$	$\frac{1-\sqrt{5}}{4}$	$\frac{3-\sqrt{5}}{2(-1+\sqrt{5})}$	$\frac{1}{1-\sqrt{5}}$	$\frac{-1}{1-\sqrt{5}}$	$\frac{-1+\sqrt{5}}{4}$
$\frac{-1}{1+\sqrt{5}}$	$\frac{-1-\sqrt{5}}{4}$	$\frac{3+\sqrt{5}}{2(1+\sqrt{5})}$	$\frac{1}{1+\sqrt{5}}$	$\frac{1}{1+\sqrt{5}}$	$\frac{-1-\sqrt{5}}{4}$
$\frac{1}{1+\sqrt{5}}$	$\frac{1+\sqrt{5}}{4}$	$\frac{-3-\sqrt{5}}{2(1+\sqrt{5})}$	$\frac{1}{1+\sqrt{5}}$	$\frac{-1}{1+\sqrt{5}}$	$\frac{-1-\sqrt{5}}{4}$
$\frac{-1}{1+\sqrt{5}}$	$\frac{1+\sqrt{5}}{4}$	$\frac{3+\sqrt{5}}{2(1+\sqrt{5})}$	$\frac{-1}{1+\sqrt{5}}$	$\frac{-1}{1+\sqrt{5}}$	$\frac{1+\sqrt{5}}{4}$
$\frac{1}{1+\sqrt{5}}$	$\frac{-1-\sqrt{5}}{4}$	$\frac{-3-\sqrt{5}}{2(1+\sqrt{5})}$	$\frac{-1}{1+\sqrt{5}}$	$\frac{1}{1+\sqrt{5}}$	$\frac{1+\sqrt{5}}{4}$
$\frac{1}{1-\sqrt{5}}$	$\frac{-1+\sqrt{5}}{4}$	$\frac{3-\sqrt{5}}{2(-1+\sqrt{5})}$	$\frac{1}{-1+\sqrt{5}}$	$\frac{1}{1-\sqrt{5}}$	$\frac{1-\sqrt{5}}{4}$
$\frac{1}{-1+\sqrt{5}}$	$\frac{1-\sqrt{5}}{4}$	$\frac{3-\sqrt{5}}{2(-1+\sqrt{5})}$	$\frac{1}{-1+\sqrt{5}}$	$\frac{-1}{1-\sqrt{5}}$	$\frac{1-\sqrt{5}}{4}$

5. 3D3A 广义 Stewart 平台的距离约束. 现在介绍怎样求解三个距离约束. 基体和平台之间有六种距离约束：**DPP**, **DPL**, **DPH**, **DLP**, **DHP**, **DLL**. 对于每一种距离约束, 比如 $C =$**DPL**, 在三个角度约束和这个距离约束下, 平台上对应的几何元 e 的集合, 即 e 上所有可能的点的集合, 称为这个约束的轨迹, 记作 L_C 或 L_{DPL}. 另外, 我们总可假设平台上的几何对象是点. 否则, 如果平台上得几何元素

是一条直线或一个平面, 我们可选取其上的一个点.

定理 6.5.6 假设 D_i 是平台上一个几何元 e_i 和基体上的一个几何元之间的一个距离约束. 如果平台的方向是确定的, 那么 e_i 的轨迹, 即 e_i 上所有可能的点的集合, 是平面, 或球面, 或圆柱面.

我们分析其中两种情况, 其他情况可以类似得到.

L_{DPP}. 令约束是 $\mathrm{DIS}(p_1, p_2) = d$. p_1 在约束下的轨迹是一个中心在 p_2, 半径为 d 的球面.

L_{DLP}. 令约束是 $\mathrm{DIS}(l, p) = d$. 如果仅考虑距离约束, 直线 l 是中心在 p 半径为 d 的球面的所有的切线. 进一步假定直线 l 的方向是固定的, 那么 l 的轨迹是一个中轴线为 l_1, 半径为 d 的圆柱面, 这里 l_1 是一条过 p 平行于 l 的直线.

定理 6.5.7 令 D_i 是一个平台上的一个几何元 e_i 和基体上的一个几何元间的距离约束. 如果平台的方向是固定的, 那么 e_i 上的任意给定点的轨迹是 L_{D_i}.

如果 e_i 是一个点, D_i 必是 **DPP**, **DPL**, **DPH** 中之一. 此时, 命题显然是正确的. 否则, e_i 是一条直线或一个平面. 根据上面的讨论, 我们知道 e_i 上点的集合是 L_{D_i}. 因此 L_{D_i} 可作为 e_i 上一个给定点的轨迹.

在加入三个角度约束后, 平台的方向就被确定了. 进一步确定平台的位置, 只需确定平台上的一个点的位置.

我们进一步假定平台的方向已知. 对于给定的平台和基体上的几何元间的三个距离约束 $D_i, i = 1, 2, 3$, $D_i, i = 1, 2, 3$ 所确定的直线或平面的方向也是确定的. 我们可以如下求得满足三个距离约束的平台的一个位置.

- 列出由 D_i 确定的 $L_i(x, y, z) = 0, i = 1, 2, 3$ 的轨迹方程.
- 令 D_i 是平台上几何元 e_i 和基体上的几何元 f_i 间的距离约束. 如果 e_i 是一个点, 令 $p_i = e_i$. 否则, 如果 e_i 是一条直线或一个平面, 选取 e_i 上的任意一个固定点作为 p_i. 令 $p_i = (x_i, y_i, z_i)$.
- 根据定理 6.5.7, 在加入距离约束 D_i 后, 点 p_i 在轨迹 L_{D_i} 上. 此外, 因为当加入每个约束 $D_i (i = 2, 3)$ 时, 平台的方向是固定的, 点 p_1 一定在轨迹 $L'_i (i = 2, 3)$ 上, $L'_i (i = 2, 3)$ 是 L_{D_i} 沿方向 $p_1 - p_i$ 的平移. 在加入三个距离约束后, 点 p_i 的位置 p'_1 是三个面 L_1, L'_2 与 L'_3 的交点：

$$\begin{aligned} &L_1(x, y, z) = 0 \\ &L_2(x + x_1 - x_2, y + y_1 - y_2, z + z_1 - z_2) = 0 \\ &L_3(x + x_1 - x_2, y + y_1 - y_2, z + z_1 - z_2) = 0 \end{aligned} \tag{6.30}$$

- 根据定理 6.5.8 求解方程组 (6.30), 我们找到点 p_1 的新位置 p'_1.

- 沿平移向量 $p_1p_1' = p_1' - p_1$ 移动平台, 它满足三个距离约束.

这里考虑的轨迹的类型为平面, 球面和圆柱面. 可以分以下几种情形添加三个距离约束. 所有的方程系统均可以用吴特征列方法求解.

定理 6.5.8 用**P**, **S** 和**C** 表示一个平面, 一个球面和一个圆柱面. 我们用三个字母的组合表示三个这样的曲面的交. 例如, **PPP** 表示三个平面的交. 如果三个曲面的交点个数有限, 我们有下面的上界. 就实解而言, 这些上界是最佳的.

- **PPP** 的情形. 对于**PPP** 的情形, 方程系统可化为三个线性方程构成的三角列. 此时, 至多有一个解.
- **PPC**, **PPS**, **SSS** 或**PSS**的情形. 对于每一种情形, 方程系统可化为两个线性方程和一个二次方程构成的三角列. 此时, 至多有两个解.
- **PCC**, **SSC** 或**PSC** 的情形. 对于每一种情形, 方程系统可化为两个线性方程和一个四次方程构成的三角列. 此时, 至多有四个解.
- **CCC** 或**SCC** 的情形. 对于每一种情形, 方程系统可化为两个线性方程和一个八次方程构成的三角列. 此时, 至多有八个解.

让我们考虑**PSC**的情形, 即找出一个平面, 一个球面和一个圆柱面的交. 平面, 球面和圆柱面的方程如下.

$$\begin{cases} d_1x + d_2y + d_3z + d_4 = 0 \\ x^2 + y^2 + z^2 + d_5x + d_6y + d_7z + d_8 = 0 \\ d_9x^2 + d_{10}y^2 + d_{11}z^2 + (d_{12}y + d_{13}x + d_{14})z \\ \qquad + (d_{15}z + d_{16})y + d_{17}x + d_{18} = 0 \end{cases} \tag{6.31}$$

不失一般性, 假定 $d_3 \neq 0$. 上面的方程系统可以利用吴特征列方法以变量序 $x < y < z$ 化为三角列形式. 在 MMP 中命令如下：

```
>p1:=d_1*x+d_2*y+d_3*z+d_4;
>p2:=x^2+y^2+z^2+d_5*x+d_6*y+d_7*z+d_8;
>p3:=d_9*x^2+d_10*y^2+d_11*z^2+(d_12*y+d_13*x+d_14)*z+
     (d_15*z+d_16)*y+d_17*x+d_18;
>ps:=[p1, p2, p3];
>vs:=[z, y, x];
>wsolve(ps, vs, [], "weak");
```

约化结果如下:

$$\begin{cases} z_6x^4 + z_7x^3 + z_8x^2 + z_9x + z_{10} = 0 \\ (z_1x + z_2)y + z_3x^2 + z_4x + z_5 = 0 \\ d_3z + d_2y + d_1z + d_4 = 0 \end{cases} \tag{6.32}$$

z_i 是关于 d_i 的多项式, 具体表达式省略.

我们仍需说明上面的多项式系统确实可以有四个实解. 对此, 我们给出一个几何证明. 一个平面和一个球面的交是一个圆 c. 如果适当的选择 c 的位置, 确实可以和圆柱面有四个交点. 对于**PPP**, **PPC**, **PPS**, **SSS**, **PSS**, **PCC**, **SSC**, **PSC**, **SCC** 的情形, 与**PSC**的情形类似, 可以证明命题中的上界是最佳的. 具体的想法是首先找出两个面的交, 得到一条直线, 或者一个圆, 或者两个圆, 再用第三个面和直线或圆相交. 例 6.5.9 表明**CCC** 的情形可以有八个实解.

例 6.5.9 继续例 6.5.5. 现在我们对图 6.8 中的广义Stewart平台加入三个距离约束. 这里我们使用同样的记号来表示基体和平台上的直线和点. 假定三个角度约束已经添加.

令三个距离约束为 $|P_1L_4| = r_1$, $|P_1L_5| = r_2$, $|P_1L_6| = r_3$. 不失一般性, 我们可以假定 $P_2 = (0, 0, 0)$. 假设在加入三个距离约束后点 P_1 的坐标变为 $P_1^* = (x, y, z)$. 根据前面列出的算法, P_1^* 是三个圆柱面的交, 并且满足下面的方程系统：

$$x^2 + y^2 - r_1^2 = 0, x^2 + z^2 - r_2^2 = 0, y^2 + z^2 - r_3^2 = 0 \tag{6.33}$$

上面的方程系统可以利用吴特征列方法以变量序 $x < y < z$ 化为三角列形式：

$$\begin{cases} r_1^2 + 2z^2 - r_2^2 - r_3^2 = 0 \\ 2y^2 - r_1^2 + r_2^2 - r_3^2 = 0 \\ 2x^2 - r_1^2 - r_2^2 + r_3^2 = 0 \end{cases} \tag{6.34}$$

很明显, 上面的方程系统有八个复解. 如果 $r_1^2 < r_2^2 + r_3^2, r_2^2 < r_1^2 + r_3^2, r_3^2 < r_1^2 + r_2^2$, 这个方程系统有八个实解. 例如, 取 $r_1 = 6$, $r_2 = 4$, $r_3 = 5$, 我们得到八个实解, 见表 6.2. 在 MMP 中, 这八个实解可以如下得到：

```
>r_1:=6; r_2:=4; r_3:=5;
>roots([r_1^2+2*z^2-r_2^2-r_3^2,2*y^2-r_1^2+r_2^2-r_3^2,
2*x^2-r_1^2-r_2^2+r_3^2],[z,y,x]);
```

表 6.2 平台上的一个点的八个实解

	1	2	3	4	5	6	7	8
x	$\frac{3\sqrt{6}}{2}$	$\frac{3\sqrt{6}}{2}$	$\frac{3\sqrt{6}}{2}$	$\frac{3\sqrt{6}}{2}$	$\frac{-3\sqrt{6}}{2}$	$\frac{-3\sqrt{6}}{2}$	$\frac{-3\sqrt{6}}{2}$	$\frac{-3\sqrt{6}}{2}$
y	$\frac{3\sqrt{10}}{2}$	$\frac{3\sqrt{10}}{2}$	$\frac{-3\sqrt{10}}{2}$	$\frac{-3\sqrt{10}}{2}$	$\frac{3\sqrt{10}}{2}$	$\frac{3\sqrt{10}}{2}$	$\frac{-3\sqrt{10}}{2}$	$\frac{-3\sqrt{10}}{2}$
z	$\frac{\sqrt{10}}{2}$	$\frac{-\sqrt{10}}{2}$	$\frac{\sqrt{10}}{2}$	$\frac{-\sqrt{10}}{2}$	$\frac{\sqrt{10}}{2}$	$\frac{-\sqrt{10}}{2}$	$\frac{\sqrt{10}}{2}$	$\frac{-\sqrt{10}}{2}$

作为定理 6.5.4 和 6.5.8 的直接结论, 我们有下面的结果.

定理 6.5.10 加入三个角度约束, 平台得方向最多有8个解. 加入三个距离约束, 平台得位置可以有1, 2, 4, 8个解. 所以, **3D3A** 广义Stewart平台根据约束的不同类型, 可以有 $2^k, k = 3, \cdots, 6$ 个解. 这些界对实解而言是最优的.

对于一些特殊形状的广义 Stewart 平台, 我们可以用上面所讲的方法给出更加精确的解的上界.

例 6.5.11 再次考虑第 11 页上的广义 Stewart平台 (图 6.9). 这里角度约束都是**ALL** 类型的, 距离约束都是**DLL**类型的. 加入三个角度约束, 平台的方向有四个解. 加入三个距离约束, 即求三个平面的交, 得到一个解. 因此这个 **3D3A** 广义Stewart平台的正解的个数是 4. 如果直接用代数方程计算, 这一问题的Bezout数是 $2,097,152$. 由此可见, 我们的方法大大化简了广义 Stewart 平台的计算问题.

例 6.5.12 考虑图 6.10 所示的 **3D3A** 广义 Stewart 平台. 这里平台是 $h_1h_2p_5$, 基体是 $h_3h_4p_6$. 如此图所示角度约束是关于 $h_2 - h_3, h_1 - h_3, h_1 - h_4$ 的. 距离约束是 $h_2 - p_6, p_5 - h_4, p_5 - p_6$. 三个角度约束有四个解. 三个距离约束可以约化为两个平面和一个球面的交, 因此有两个解. 这个 **3D3A** 广义 Stewart 平台的正解的个数是 8.

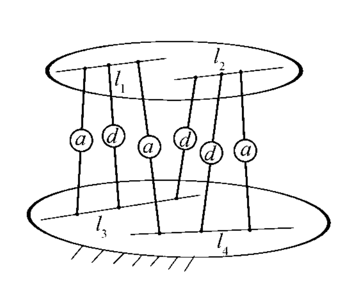


图6.9 一个特殊的**3D3A**GSP

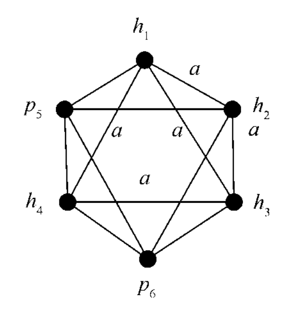


图6.10 一个特殊的**3D3A**GSP

6.5.3 曲面拼接

考虑 CAGD(计算机辅助几何设计) 中的曲面拼接问题.

问题 SF 给定 $\mathbb{R}^3$ 空间三条实不可约代数曲线 C_i, C_j, C_k, 其中 $i \in I, j \in J, k \in K$, I, J, K 是有限集. 同时给定两个分别包含 C_j, C_k 的实不可约代数曲面 $S_j, S_k, (j \in J, k \in K)$. 确定一个具有给定幂次 m 的实不可约代数曲面 S, 满足下述条件:

- 对 $i \in I, j \in J, k \in K$, S 包含所有曲线 C_i, C_j, C_k.
- 对 $j \in J$, $k \in K$, S 分别沿曲线 C_j, C_k 与 S_j, S_k 光滑拼接. 更准确地说, 对 C_j

或 C_k 上每一关于 C_j, S_j 或 C_k, S_k 的正则点, 所求曲面 S 在此点也是正则的. 进一步, S 与 S_j 或 S 与 S_k 在此点将具有相同切平面.

- 对任一 $k \in K$, S 沿曲线 C_k 具有与 S_k 相同曲率. 准确地说, 对 C_k 上每个关于 C_k, S_k 的正则点, 曲面 S 与 S_k 在此点不仅具有相同切平面而且具有相同的 (Gaussian) 曲率.

为了解决问题 SF, 首先引入某些基本定理. 我们考虑一般实空间 $\mathbb{R}^3$ 中的实曲线及曲面. 我们用通常的 x, y, z 代替 x_1, x_2, x_3, 变量序为 $x \prec y \prec z$. 那么, $\mathbb{R}^3$ 中的实不可约代数曲面将可由方程 $P = 0$ 定义, 其中 P 是一实多项式, 在 $\mathbb{C}$ 中不可约, 且 $P = 0$ 具有实维数为 2 的实轨迹. P 的主变元是 x, y 或 z. 如果 P 的主变元是 x, 那么 P 必是 x 的线性方程, 此时曲面是某平面.

$\mathbb{R}^3$ 中实不可约代数曲线将由一不可约升列 $\mathcal{A} = \{P_1, P_2\}$ 定义, 其中 $P_1, P_2 \in \mathbb{R}[x, y, z]$. P_1, P_2 的主变元是 x, y 或 x, z 或 y, z.

假设 C 是一不可约实代数曲线, 由不可约升列 $\mathcal{A} = \{P_1, P_2\}$ 确定. 其中 $P_1, P_2 \in \mathbb{R}[x, y, z]$, S, S' 是分别由 $P = 0$ 及 $P' = 0$ 定义的不可约代数曲面, $P, P' \in \mathbb{R}[x, y, z]$. 我们有下面的定理.

定理 6.5.13 曲面 S 包含整条曲线 C 当且仅当

$$\operatorname{prem}(P, \mathcal{A}) = 0$$

定理 6.5.14 假设 C 完全包含在 S 及 S' 中, 而并未包含在 S 或 S' 的奇异轨迹中. 取 $P_x = \dfrac{\partial P}{\partial x}, P'_x = \dfrac{\partial P'}{\partial x}$ 等等, 且建立下述多项式

$$\left.\begin{aligned} D_1 &= P_x * P'_y - P_y * P'_x \\ D_2 &= P_x * P'_z - P_z * P'_x \\ D_3 &= P_y * P'_z - P_z * P'_y \end{aligned}\right\}$$

那么 S 及 S' 在 C 上所有的对 C, S, S' 都正则的点处沿 C 光滑拼接当且仅当

$$\operatorname{prem}(D_i, \mathcal{A}) = 0, \quad i = 1, 2, 3$$

对由 $P = 0$ 定义的实不可约代数曲面, 在正则点的曲率 κ 如下:

$$\kappa = \frac{V}{H^2}$$

其中

$$H = \sqrt{P_x^2 + P_y^2 + P_z^2}$$

$$V = Cyc(P_{xx} * P_{yy} * P_z^2 - 2 * Cyc(P_x * P_y * P_{xy} * P_{zz}))$$
$$+2 * Cyc(P_x * P_y * P_{xz} * P_{yz}) - Cyc(P_x^2 * P_{yz}^2)$$

其中 Cyc 表示相应表达式关于变量 x, y, z 及 $P_x, P_{xx} = \dfrac{\partial^2 P}{\partial x^2}$ 的循环和, 它们均取值于正则点.

定理 6.5.15 令 C, S, S' 满足定理 6.5.14 中的条件, 如前所述. 那么, S, S' 在 C, S, S' 的正则点上具有相同曲率当且仅当

$$\text{prem}(H^2 * V' - H'^2 * V, \mathcal{A}) = 0$$

其中 H' 及 V' 是关于 P' 的多项式, 类似于 H 及 V 是关于 P 的多项式.

根据定理6.5.13~6.5.15, 解决问题 SF 将转化成验证余式是否为 0, 而这可被进一步化简为多项式方程的求解, 如下所示.

对由 $P = 0$ 定义的幂次为 m 的不可约代数曲面, 把 P 写成下述形式

$$P = \sum_m u_{abc} * x^a * y^b * z^c \tag{6.35}$$

其中的 $\sum_m$ 是在满足下述条件的三元组 (a, b, c) 上求和：

$$0 \leqslant a \qquad 0 \leqslant b \qquad 0 \leqslant c \qquad a + b + c \leqslant m \tag{6.36}$$

我们把满足 (6.36) 的所有 u_{abc} 的集合记成 $\mathbb{U}_m$.

例如考虑 S 是否包含整条由不可约升列 $\mathcal{A}$ 定义的不可约代数曲线 C 的问题. 由定理6.5.13, 等价于考虑下述条件

$$R = \text{prem}(P, \mathcal{A}) = 0 \tag{6.37}$$

记 R 中变量 x, y, z 的不同幂次的系数构成的多项式集是 $US_0 \subset \mathbb{R}[\mathbb{U}_m]$. 那么条件 (6.37) 等同于下述条件

$$US_0 = 0 \tag{6.38}$$

为了满足问题 SF 中的条件 1, 与 (6.38) 中 US_0 相似, 我们对每条曲线 C_i, C_j, C_k 构造多项式集 $US_{ai}, US_{aj}, US_{ak}$. 类似的, 对问题 SF 中的要求 2, 由定理 6.5.14 可得多项式集 $US_{bi}, US_{bj}, US_{bk}$, 对问题 SF 中的要求 3, 由定理 6.5.15 可得多项式集 US_{ck}, 它们的零点是将要证明的相应的条件. 注意到 $US_{ai}, US_{aj}, US_{ak}, US_{bj}, US_{bk}$ 关于 u' 是线性的, 而 US_{ck} 关于 u 则一般非线性. 把上述所有多项式集合并成一多项式集 $US \subset \mathbb{R}[\mathbb{U}_m]$, 所有满足要求的问题 SF 的解 S 由实零点集 $\text{Zero}_{\mathbb{R}}(US_0)$ 给出.

下面举例说明. $\mathbb{R}^3$ 中一般三次曲面的形式为

$$\begin{aligned}
&f(x,y,z)\\
=&u_{300}*z^3+z^2(u_{210}*y+u_{201}*x+u_{200})+z*(u_{120}*y^2\\
&+u_{111}*y*x+u_{102}*x^2+u_{110}*y+u_{101}*x+u_{100})+u_{030}*y^3\\
&+u_{021}*y^2*x+u_{012}*y*x^2+u_{003}*x^3+u_{020}*y^2\\
&+u_{011}*y*x+u_{002}*x^2+u_{010}*y+u_{001}*x+u_{000}\\
=&0
\end{aligned} \tag{6.39}$$

下面, 我们将举例表明如何采用上述的方法来确定一个三次曲面使其满足问题 SF 中的关于 1, 2, 3 的精确陈述中的某些要求.

例 6.5.16 在 $\mathbb{R}^3$ 中考虑以 x, y 轴为轴心线的两圆柱 CYL_1, CYL_2 及由平面截得的与此两轴分别垂直的两圆部分 C_1, C_2. 那么这些圆形部分将由升列 $AS_1 = \{C_{11}, C_{12}\}$ 及 $AS_2 = \{C_{21}, C_{22}\}$ 定义, 其中

$$\begin{aligned}
C_{11} &= x-d_1, \qquad & C_{12} &= z^2+y^2-r_1^2\\
C_{21} &= y-d_2, \qquad & C_{22} &= z^2+x^2-r_2^2
\end{aligned}$$

而两圆柱则分别由 $C_{12}=0, C_{22}=0$ 定义. 很自然地, 我们假设 d_1, d_2, r_1, r_2 非 0, 且为正的. 现在我们确定一个三次曲面 (6.39), 使其包含圆 C_1, C_2 且沿圆与圆柱 CYL_1, CYL_2 光滑相接. 由上述方法得 35 个方程, 用 MMP 计算, 易得到这样的解存在当且仅当

$$r_1^2+d_1^2=r_2^2+d_2^2 \tag{6.40}$$

在 MMP 中计算过程如下:

```
>pset:=[a5-a3, -a19-d1*a15-a5*d1^2-3*a3*r1^2+2*r1^2*a7, -3*a7+3*a3,
-a13+a11-d2*a9+d2*a4, a20+a13*r2^2+d2^3*a2+d2^2*a12+d2*a18
+d2*a9*r2^2, a20+r1^2*a13+d1^3*a1+d1^2*a11+d1*a17+d1*r1^2*a8,
a14+2*d2*a6, a10, a2-a9, a1-a8, a18+r1^2*a9+d1^2*a4+d1*a14,
a19+d1*a15+a5*d1^2+a3*r1^2, a12-a13+d1*a6-d1*a8, a6-a8, -a9+a4,
a7-a3, a16+d1*a10, a17+r2^2*a8+d2^2*a6+d2*a14,
d2*a16+a19+a7*d2^2+a3*r2^2, d2*a10+a15, a14+2*d1*a4,
a17+r1^2*a8+3*d1^2*a1+2*d1*a11, a15+2*a5*d1, a16+2*a7*d2,
-2*a16-2*d1*a10, 3*a2-3*a9, d2*r2^2*a10+a15*r2^2, -3*a5+3*a3,
-2*d2*a10-2*a15, 3*a1-3*a8, r1^2*a16+d1*r1^2*a10,
a18+a9*r2^2+3*d2^2*a2+2*d2*a12, 2*a12-2*a13+2*d1*a6-2*d1*a8,
-2*a13+2*a11-2*d2*a9+2*d2*a4,
```

```
-d2*a16-a19-a7*d2^2-3*a3*r2^2+2*a5*r2^2];
>ord:=[a20, a19, a18, a17, a16, a15, a14, a13, a12, a11, a10, a9,
a8, a7, a6, a5, a4, a3, a2, a1, d2, r2, d1, r1];
>wsolve(pset, ord, [r1, r2, d1, d2], "weak");
```

在此情形, 可能的三次曲面如下给出:

$$\begin{aligned}&z^2*(y*d^2+x*d_1-d_2^2-d_1^2)+(y^3*d_2+x^3*d_1)+y*x*(y*d_1\\&+x*d_2-2*d_2*d_1)-(y^2+x^2)*(d_2^2+d_1^2)+y*d_2*(d_1^2-r_1^2)\\&+x*d_1*(d_2^2-r_2^2)+(r_1^2+d_1^2)*(r_2^2+d_2^2)-2*d_1^2*d_2^2=0\end{aligned} \tag{6.41}$$

见图 6.11.

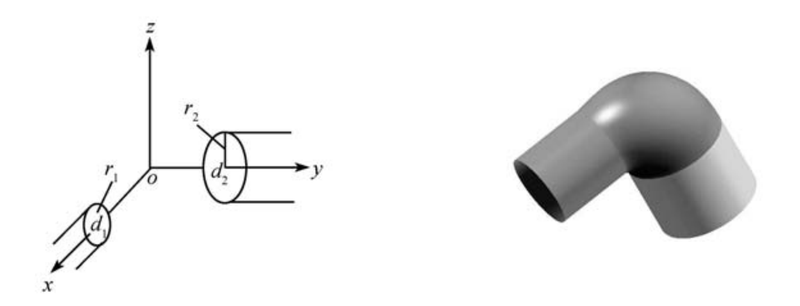


图 6.11　两个管道的拼接

6.5.4　单代数簇有理表示的隐式化

一个代数簇可以表示为各种形式, 最基本的表示即 *隐式表示*, 或 *生成元表示*. 例如, 单位圆可以隐式表示如下：

$$x^2+y^2-1=0$$

代数簇另一种常见的表示是 *代数簇的有理参数表示*. 例如, 单位圆可以有理参数表示如下：

$$x=\frac{1-t^2}{1+t^2},\quad y=\frac{2t}{1+t^2}$$

这里 t 为参数. 由此自然导致两类问题：

- *有理参数化问题* 是指求隐式代数簇的有理参数表示.
- *隐式化问题* 是指有理参数化的逆问题, 即从参数方程求相应的隐式方程.

有理参数化与隐式化是构造性代数几何的基本问题之一. 在应用方面, 曲面曲线的参数化与隐式化是几何建模和计算机图形学的重要研究内容. 这是因为, 使用代数

曲面的有理表示可以很容易生成相应的曲面, 而使用隐式表示则更容易判定一个点与曲面的位置关系. 所以, 在对一个曲面进行操作时, 我们希望同时具有两种表示.

6. 隐式代数簇的计算. 本节介绍怎样用 MMP 实现有理参数方程的隐式化. 设 $P_1,\cdots,P_n, Q_1,\cdots,Q_n$ 为 $\mathbb{K}[u_1,\cdots,u_m]$ 或 $\mathbb{K}[\mathbb{U}]$ 中的非零多项式. 我们称

$$x_1=\frac{P_1(\mathbb{U})}{Q_1(\mathbb{U})},\cdots,x_n=\frac{P_n(\mathbb{U})}{Q_n(\mathbb{U})} \tag{6.42}$$

是一个*有理参数表示*. 我们假定不是所有的 $P_i(\mathbb{U})$ 和 $Q_i(\mathbb{U})$ 都在 $\mathbb{K}$ 中, 并且 $\gcd(P_i(\mathbb{U}), Q_i(\mathbb{U}))=1$.

有理参数表示 (6.42) 的*隐式理想*定义为

$$\mathbf{ID}(P,Q)=\{P\in\mathbb{K}[x_1,\cdots,x_n]|P(P_1/Q_1,\cdots,P_n/Q_n)\equiv 0\}$$

称 $\mathrm{Zero}(\mathbf{ID}(P,Q))$ 为 (6.42) 的*隐式簇*. 设 $\mathcal{V}$ 是一个代数簇, 如果 $\mathcal{V}$ 是 (6.42) 的隐式簇, 则称 $\mathcal{V}$ 是*单有理的*, 而 (6.42) 是 $\mathcal{V}$ 的一个*有理参数表示*. 单有理代数簇是一种结构简单的代数簇. 不难证明: $\mathbf{ID}(P,Q)$ *是一个素理想*, $\eta=(P_1/Q_1,\cdots,P_n/Q_n)$ *是它的一个母点*.

设 $\mathbb{P}=\{P_1-x_1Q_1,\cdots,P_n-x_nQ_n\}$, $\mathbb{D}=\{Q_1,\cdots,Q_n\}$. 因为 $\mathbb{P}$ 在秩 $u_1<\cdots<u_m<x_1<\cdots<x_n$ 下是一个升列, 可以定义 $\mathrm{sat}(\mathbb{P})$. 我们有

引理 6.5.17 *$\mathrm{sat}(\mathbb{P})$ 是一个素理想, 它的维数是 m; $\mathrm{sat}(\mathbb{P})\cap\mathbb{K}[\mathbb{X}]$ 是 (6.42) 的隐式理想.*

为了计算隐式簇, 一个关键的事实是 $\mathrm{Zero}(\mathbb{P}/\mathbb{D})=\mathrm{Zero}(\mathrm{sat}(\mathbb{P})/\mathbb{D})$. 由此可知, 为了计算隐式理想的特征列, 我们只需在秩 $x_1<\cdots<x_n<u_1<\cdots<u_m$ 下计算 $\mathrm{Zero}(\mathbb{P}/\mathbb{D})$ 的一个零点分解. 利用吴零点分解定理有

$$\mathrm{Zero}(\mathbb{P}/\mathbb{D})=\cup_i\mathrm{Zero}(\mathrm{sat}(\mathcal{A}_i)/\mathbb{D})$$

在上面的分解中, 存在着一个 $\mathcal{A}_k$, 使得 $\mathrm{sat}(\mathcal{A}_k)=\mathrm{sat}(\mathbb{P})$. 通过重新命名变量, 我们可以把 $\mathcal{A}_k$ 写成下面的形式：

$$\begin{array}{l} A_1(x_1,\cdots,x_{d+1}) \\ \quad\cdots\cdots\cdots\cdots \\ A_{n-d}(x_1,\cdots,x_n) \\ B_1(x_1,\cdots,x_n,t_1,\cdots,t_{s+1}) \\ \quad\cdots\cdots\cdots\cdots \\ B_{m-s}(x_1,\cdots,x_n,t_1,\cdots,t_m) \end{array} \tag{6.43}$$

其中 $d+s=m$. 由上面讨论, 有以下结论：

定理 6.5.18 参数方程 (6.42) 的隐式理想是 $\mathbf{ID}(P,Q)=\mathrm{sat}(A_1,\cdots,A_{n-d})$.

例 6.5.19 考虑参数方程：

$$x=u+v,\ y=u^2+v^2+2uv-1,\ z=u^3+v^3+3u^2v+3v^2u+1 \tag{6.44}$$

令 $\mathbb{P}=\{x-u-v,y-u^2-v^2-2uv+1,z-u^3-v^3-3u^2v-3v^2u-1\}$. 在变量序 $x<y<z<u<v$ 下, 对 $\mathbb{P}$ 进行零点分解. 在MMP中命令如下：

```
>ps:=[x-u-v, y-u^2-v^2-2*u*v+1, z-u^3-v^3-3*u^2*v-3*v^2*u-1];
>vs:=[v, u, z, y, x];
>wsolve(ps, vs, [], "weak");
```

结果为 $[[x-u-v,x^3-z+1,x^2-y-1]]$

根据以上计算结果, 我们有 $\mathrm{Zero}(\mathbb{P})=\mathrm{Zero}(\mathrm{sat}(\mathcal{A}_1))$, 其中

$$\mathcal{A}_1=\{x-u-v,x^3-z+1,x^2-y-1\}$$

由 $\mathcal{A}_1$ 的特殊形式, 我们有 $\mathrm{sat}(\mathcal{A}_1)=(\mathcal{A}_1)$. 由定理 6.5.18, (6.44) 定义了曲线 $\mathrm{Zero}(x^3-z+1,x^2-y-1)$.

一个有趣的现象是参数方程 (6.44) 有两个参数, 却定义了一条曲线. 这说明参数方程 (6.44) 中的两个参数是不独立的.

7. 独立参数问题. 如果(6.42)的隐式理想是 m 维的, 我们就称(6.42)的参数 $u_1,\cdots,u_m$ 是独立的. 或者等价的说, $\mathbb{K}(P_1/Q_1,\cdots,P_n/Q_n)$ 在 $\mathbb{K}$ 上的超越次数是 m. 我们知道素理想的维数等于其特征列的参数的个数. 由定理 6.5.18, $\mathbf{ID}(P,Q)$ 的维数为 $d=m-s$. 因此, 我们有

定理 6.5.20 假设我们已经构造了 (6.43), 那么 $\mathbf{ID}(P,Q)$ 的维数是 $d=m-s>0$. 因此, 参数是独立的, 当且仅当 $s=0$.

如果 (6.42) 的参数不独立, 我们可以找到新的有理参数方程

$$x_1=\overline{P}_1/\overline{Q}_1,\cdots,x_n=\overline{P}_n/\overline{Q}_n \tag{6.45}$$

它和 (6.42) 有相同的隐式簇, 但它的参数是独立的. 基本想法如下：设 J 是 (6.43) 的初式乘积. 因为 (6.43) 是不可约的, 我们可以求得对 (6.43) 约化并且不含有 (6.43) 中主变元的非零多项式 K 和 q_i, 使得

$$K\in\mathrm{Ideal}(A_1,\cdots,A_{n-d},B_1,\cdots,B_{m-s},J)$$
$$q_i\in\mathrm{Ideal}(A_1,\cdots,A_{n-d},B_1,\cdots,B_{m-s},Q_i)$$

设 $M=K\cdot\prod_{j=1}^n q_j$. M 不在 sat(6.43) 中. 我们把 M 中的 x_i 用 P_i/Q_i 代替, 可以得到一个关于 u_i 的非零多项式 N. 在 $\mathbb{K}$ 中存在 $h_1,\cdots,h_s$, 使得当 u_i 用 h_i,

$i=1,\cdots,s$ 代替时, N 变为一个非零多项式 $\overline{N}$. 设 $\overline{P}_i,\overline{Q}_i$ 是由 h_i 代替 P_i,Q_i 中的 $u_i,i=1,\cdots,s$ 得到的多项式. 显然, $\overline{P}_i,\overline{Q}_i$ 只包含 $u_{s+1},\cdots,u_m$. 这样, 我们就得到了 (6.45), 而且可以证明 (6.45) 满足我们所要求的条件.

例 6.5.21 在例 6.5.19 中我们有 $d=1,s=1$. 因此, 参数 u 和 v 不独立. 重新参数化 (6.44) 需要计算 M. 由于 $\mathrm{prem}(u-v,\mathcal{A})=2u$, $M=2(x+1)u$. 选取 u 的一个值 (比如 1), 使得 M 不为 0. 我们可以得到参数方程

$$x=\frac{v+1}{1-v},\ y=\frac{2v^2+2}{(1-v)^2},\ z=\frac{2v^3+6v}{(1-v)^3} \tag{6.46}$$

令 $\mathbb{P}=\{x(1-v)-v-1,y(1-v)^2-2v^2-2,z(1-v)^3-2v^3-6v\}$. 在变量序 $x<y<z<v$ 下, 对 $\mathbb{P}$ 进行零点分解. 在MMP中命令如下：

```
>ps:=[x*(1-v)-v-1, y*(1-v)^2-2*v^2-2, z*(1-v)^3-2*v^3-6*v];
>vs:=[v, z, y, x];
>wsolve(ps, vs, [1-v], "weak");
```

结果如下：$[[vx-x+v+1,x^3-z-1,x^2-y+1]]$

我们得到得方程 (6.46) 与原方程 (6.44) 有相同的隐式理想和独立参数 v.

8. 逆映射和正则参数方程. 所谓的逆问题就是给定一个 (6.42) 映像中的点 $(a_1,\cdots,a_n)$, 计算 $\tau=(\tau_1,\cdots,\tau_m)$, 使得 $a_i=P_i(\tau)/Q_i(\tau),i=1,\cdots,n$. 这个问题可以转变为方程求解问题. 下面将说明, 在某些情况下, 我们可以找到逆问题的闭形式解.

逆问题是和参数方程是不是正则的紧密相关. 我们称 (6.42) 是正则的, 如果对于隐式簇的一个母点 $(a_1,\cdots,a_n)$, 仅仅有一个 $(\tau_1,\cdots,\tau_m)\in\mathbb{E}^m$, 使得 $a_i=P_i(\tau_1,\cdots,\tau_m)/Q_i(\tau_1,\cdots,\tau_m)$, $i=1,\cdots,n$.

假设 (6.42) 的参数 $u_1,\cdots,u_m$ 是独立的, 即 $s=0$. 那么 (6.43) 就变为

$$\begin{aligned}&A_1(x_1,\cdots,x_{m+1}),\cdots,A_{n-m}(x_1,\cdots,x_n),\\&B_1(x_1,\cdots,x_n,u_1),\cdots,B_m(x_1,\cdots,x_n,u_1,\cdots,u_m)\end{aligned} \tag{6.47}$$

我们有

定理 6.5.22 使用上面的符号. (6.47) 是正则的, 当且仅当 $B_i=I_iu_i-U_i$ 关于 u_i 是线性的. 在这种情况下, 逆映射是

$$u_1=U_1/I_1,\cdots,u_m=U_m/I_m$$

其中 I_i 和 U_i 是 $\mathbb{K}[\mathbb{X}]$ 中的多项式.

如果有理参数方程是正则的, 则有理参数方程本身与其逆映射就提供了一个参数方程的隐式簇与 m 维仿射空间之间的双有理映射. 与仿射空间双有理等价的代

数簇称为 *有理代数簇*. 判定一个单有理代数簇是否有理的问题还未解决. 下面定理给出了代数曲线情形这一问题的解答.

定理 6.5.23 *如果 $m=1$ 且 (6.42) 非正则, 我们可以得到参数 $s=f(u_1)/g(u_1)\in\mathbb{K}(u_1)$, 使得 (6.42) 可以用 s 重新参数化, 而且重新参数化后的参数方程是正则的.*

$$x_1=\frac{F_1(s)}{G_1(s)},\ \cdots,\ x_n=\frac{F_n(s)}{G_n(s)}$$

具体做法如下. 由 (6.47), 我们可以找到曲线的逆映射 $B_1(x_1,\cdots,x_n,u_1)=0$. B_1 是 x 和 u_1 之间次数最低的关系. 换言之, $B_1'(y)=B_1(P_1/Q_1,\cdots,P_n/Q_n,y)=0$ 为 $\mathbb{K}'[y]$ 上的满足 $B_1'(u_1)=0$ 的最小次数多项式. $B_1'(y)$ 就是我们要计算的 $f(y)$. 故可如下得到 s：如果 B_1 关于 u_1 为线性的, 则 (6.42) 正则. 可以取 $s=u_1$. 否则, 令

$$B_1=b_ru_1^r+\cdots+b_0$$

其中 $b_i\in\mathbb{K}[x]$. 由 (6.42), b_i 可以表示为有理函数 $a_i(u_1)$, $i=0,\cdots,r$. 至少存在一个 a_i/a_r (记为 a_0/a_r) 不在 $\mathbb{K}$ 中. 令 $s=a_0/a_r$. 消去 (6.42) 和 a_rs-a_0 中的 u_1, 我们可以得到所需的参数方程. 注意到 a_i 是将 b_i 中的 x_j 替换为 P_j/Q_j $(j=1,\cdots,n)$ 得到的结果, 我们有 $s=b_0/b_r$ 是新参数方程的逆映射.

例 6.5.24 *考虑Bézier参数曲线方程：*

$$x=\frac{8s^6-12s^5+32s^3+24s^2+12s}{s^6-3s^5+3s^4+3s^2+3s+1},y=\frac{24s^5+54s^4-54s^3-54s^2+30s}{s^6-3s^5+3s^4+3s^2+3s+1}\quad(6.48)$$

令 $\mathbb{P}=\{(s^6-3s^5+3s^4+3s^2+3s+1)x-(8s^6-12s^5+32s^3+24s^2+12s),(s^6-3s^5+3s^4+3s^2+3s+1)y-(24s^5+54s^4-54s^3-54s^2+30s)\}, D=s^6-3s^5+3s^4+3s^2+3s+1$. 在变量序 $x<y<s$ 下, 计算 $\mathrm{Zero}(\mathbb{P}/D)$ 的零点分解. 在MMP中命令如下：

```
>ps:=[(s^6-3*s^5+3*s^4+3*s^2+3*s+1)*x-(8*s^6-12*s^5+32*s^3
+24*s^2+12*s), (s^6-3*s^5+3*s^4+3*s^2+3*s+1)*y-
(24*s^5+54*s^4-54*s^3-54*s^2+30*s)];
>D:=s^6-3*s^5+3*s^4+3*s^2+3*s+1;
>vs:=[s, y, x];
>wsolve(ps, vs, [D], "weak");
```

根据分解结果, 我们有 $\mathrm{Zero}(\mathbb{P}/D)=\mathrm{Zero}(\mathrm{sat}(\mathcal{A}))$, 其中 $\mathcal{A}=\{A_1,A_2\}$.

$A_1 = 224y^3+(-2268x+7632)y^2+(-54x^2-1512x-480384)y+34263x^3-424224x^2+1200960x$

$A_2 = (15273x^2+1098792x-9767808)s^2+(7280y^2+(-27006x-125592)y-174069x^2+598788x-9767808)s-7280y^2+(27006x+125592)y+189342x^2+500004x$

由于 (6.48) 定义一个平面曲线, (6.48) 是曲线 $A_1=0$ 的非正则参数方程. 为求 $A_1=0$ 的正则参数方程, 选定新的参数

$$t_1=\frac{(7280y^2+(-27006x-125592)y-174069x^2+598788x-9767808)}{(15273x^2+1098792x-9767808)}=\frac{s^2+1}{1-s}$$

消去 s, 我们有

$$x=\frac{8t_1^3+12t_1^2-36t_1+16}{t_1^3+3t_1^2-3t_1},\quad y=\frac{-24t_1^2+78t_1-54}{t_1^3+3t_1^2-3t_1}$$

容易判定这是 $A_1=0$ 的一个正则参数方程表示.

9. 有理参数表示的像与正规有理参数方程. 参数方程 (6.42) 在 $\mathbb{E}^n$ 中的 像 是

$$\mathbf{IM}(P,Q)=\{(\eta_1,\cdots,\eta_n)\in\mathbb{E}^n|\exists\tau\in\mathbb{E}^m(\eta_i=P_i(\tau)/Q_i(\tau))\}$$

显然

$$\begin{aligned}\mathbf{IM}(P,Q)&=\{(x_1,\cdots,x_n)|\exists\tau\in\mathbb{E}^m(Q_i(\tau)x_i-P_i(\tau)=0\wedge Q_i(\tau)\neq 0)\}\\&=\mathrm{Proj}_{u_1,\cdots,u_m}\mathrm{Zero}(\mathbb{P}/\mathbb{D})\end{aligned}$$

利用投影定理 4.5.4, 我们可以得到 $\mathbb{K}[\mathbb{U}]$ 中的多项式集合 $\mathcal{S}_i$ 和多项式 $d_i, i=1,\cdots,t$, 使得

$$\mathbf{IM}(P,Q)=\cup_{i=1}^t\mathrm{Zero}(\mathcal{S}_i/d_i)$$

例 6.5.25 计算下列参数方程的映像.

$$x=\frac{u^2-v^2+1}{u^2+v^2-1},y=\frac{2vu}{u^2+v^2-1},z=\frac{2u}{u^2+v^2-1}$$

令 $\mathbb{P}=\{(u^2+v^2-1)x-u^2+v^2-1,(u^2+v^2-1)y-2uv,(u^2+v^2-1)z-2u\}$, $\mathbb{D}=\{u^2+v^2-1\}$. 使用投影定理, 对 $\mathbb{P},\mathbb{D}$ 构成的拟代数簇进行投影运算, 在MMP中命令如下:

```
>ps:=[(u^2+v^2-1)*x-u^2+v^2-1,(u^2+v^2-1)*y-2*u*v,(u^2+v^2-1)*z-2*u];
>ds:=[u^2+v^2-1];
>proj(ps, ds, [v, u, x, y, z], [v, u]);
```

根据运算结果, 我们得到

$$\mathrm{Proj\ Zero}(\mathbb{P}/\mathbb{D})=\mathrm{Zero}(x^2+y^2-z^2-1/z(x+1))\cup\mathrm{Zero}(z,y,x+1)$$

将上面表示化为下面的规范形式:

$\mathrm{ProjZero}(\mathbb{P}/\mathbb{D})=\mathrm{Zero}(z^2-y^2-x^2+1)-(\mathrm{Zero}(z,y^2+x^2-1/x+1)\cup\mathrm{Zero}(x+1,z^2-y^2/y))$.

从上面的例子可以看到, 有理参数方程的像一般不等于其隐式簇. 有理参数方程 (6.42) 称为 *正规参数方程*, 如果 $\mathbf{IM}(P,Q)$ 是一个不可约代数簇. 换言之, (6.42) 可以取到其隐式代数簇上的所有点.

由于已经知道怎样计算 $\mathbf{IM}(P,Q)$, 我们可以判定形如 (6.42) 的参数方程是否是正规的. 下面给出一个对于曲线的简单判定标准.

定理 6.5.26 令 $y_1 = u_1(t)/v_1(t), \cdots, y_n = u_n(t)/v_n(t)$ 是一个代数曲线的参数方程组, 如果对某个 i 有 $\deg(u_i) > deg(v_i)$, 则它们是正规参数方程.

如果参数方程不是正规的, 我们自然想问能否找到具有相同隐式代数簇的正规参数方程. 这一问题尚未完全解决. 对于曲线情形可以处理如下. 假设

$$y_1 = u_1(t)/v_1(t), \cdots, y_n = u_n(t)/v_n(t)$$

是代数曲线的有理参数表示. 不妨设这一表示不是正规的, 则有 $\deg(u_i) \leqslant \deg(v_i)$, $i = 1, \cdots, n$. 不妨设 $d_1 = \deg(v_1) > 0$, α 是 $v_1 = 0$ 的根但不是 $u_1 = 0$ 的根. 做变换 $t = \frac{\alpha s+1}{s}$, 我们有

$$y_1 = u_1(t)/v_1(t) = u_1\Big(\frac{\alpha s + 1}{s}\Big)/v_1\Big(\frac{\alpha s + 1}{s}\Big) = \bar{u}_1(s)/\bar{v}_1(s)$$

化简后不难发现 $\deg(\bar{u}_1) = d_1 > \deg(\bar{v}_1)$. 由定理 6.5.26, 新的有理参数方程为正规. 所以, 在复数域上代数曲线的有理参数表示都可以化为正规有理表示.

例 6.5.27 单位圆的下面参数表示不是正规的. 点 $(0,1)$ 不能由这一方程给出.

$$x = \frac{2t}{t^2+1}, \quad y = \frac{t^2-1}{t^2+1}$$

为了得到正规表示, 令 $t = \dfrac{\mathbf{i}s+1}{s}$. 新的参数方程为

$$x = \frac{2s(\mathbf{i}s+1)}{2\mathbf{i}s+1}, \quad y = \frac{(is+1)^2 - s^2}{2\mathbf{i}s+1}$$

其中 $\mathbf{i} = \sqrt{-1}$. 上述方程是单位圆在复数域上的正规参数表示.

文献说明

本章所采用的代数方程求解的基本原理由吴文俊提出 ([101]). 这一算法的详细描述、改进与众多的应用见 [93],[87], [86], [23], [66], [78] 和 [105].

预解式的算法与应用来自 [42]. 含参数的方程组的求解的算法见 [37] 和 [40]. 参数方程解的个数的计算来自 [38]. 多项式方程组的局部数值解的算法见 [44]. 多项式方程组的全局数值解的算法见 [108].

本章关于串连机器人、Puma 型机器人的逆问题求解的内容来自 [87]. 关于广义 Stewart 平台及其正解的求解的内容来自 [39] 和 [43]. 关于曲面拼接的内容来自 [84], [87] 和 [83]. 关于单代数簇的有理表示的隐式化的内容来自 [33],[34] 和 [41].

第七章　代数微分方程求解

微分方程是代数方程一种质的推广, 如果说代数方程主要用于描述静态性质, 那么微分方程则主要用于描述动态性质. 微分方程的求解要比代数方程困难的多. 例如, 即使是对最简单的解的类型：常微方程的多项式解, 目前也还没有完全的方法. 另一方面, 数学机械化理论为微分方程解的结构提供了一种完备的描述方法. 我们可以以此为基础开展很多应用研究, 包括微分几何与力学中的定理证明, 某些特殊类型微分方程与微分方程某些特殊类型解的求解等. 本章将介绍怎样用 MMP 求解微分方程.

§7.1　代数微分方程求解的吴消元法

1. 问题的背景. 首先说明, 我们这里所讲的微分方程是广义的. 我们将考虑第四章中引入的具有任意变元个数的微分多项式所定义的方程.

考虑下面例子. 在 §5.4 讨论了由 Kepler 行星运动定律推导 Newton 万有引力定律, 基本想法是将 Kepler 定律用代数与微分方程表示：

$$k_1 = r - p - e * x, k_2 = x * y'' - y * x'', r^2 = x^2 + y^2, a^2 = x''^2 + y''^2 \tag{7.1}$$

我们希望由上述关系推导变量 a 与 r 之间的关系. 如第五章所讲, 对 (7.1) 中六个代数与微分多项式做吴特征列方法, 可以得到其中的一个方程如下：

$$2 * a * r' + a' * r = 0 \tag{7.2}$$

求得该微分方程的解：$a * r^2 = c$, 这里 c 为常数, 即 Newton 定律：*行星的加速度与行星到太阳的距离平方成反比*. 这样我们就自动发现了 Newton 万有引力定律.

将上述过程总结如下：

1. 用代数或微分方程 $\mathbb{P} = 0$ 表示某些物理现象.
2. 确定若干变量 U. 首先使用吴特征列方法, 我们可以得到变量 U 满足的一个或若干个微分方程 $F = 0$.
3. 最后, 我们希望知道变量 U 是否满足某些简单的关系, 例如代数关系. 换句话讲, 我们需要求 $F = 0$ 的代数解.

这里有两点需要注意. 首先, 上述过程的第二、三步都可以看作某种“微分方程求解”. 只不过第三步是明确求解微分方程, 而第二步则是由一个微分方程集合推导出若干给定变量所满足的关系. 其次, 微分方程 (7.2) 具有两个变量, 与通常的微分方程不同, 其解是一个一维的微分代数簇, 也可以称其为一个 *微分曲线*. 而且, 我们需要求方程 (7.2) 的代数解, 等价的, 我们需要知道一个微分曲线的通解是否是某类代数曲线?

注释 7.1.1 即使是形如 (7.2) 的微分方程的代数解问题也还没有解决. 考虑下面形式的自治微分方程:

$$\frac{dx}{dt} = P(x,y), \quad \frac{dy}{dt} = Q(x,y)$$

这里 $P, Q \in \mathbb{C}[x,y]$. 它对应于下面自治的平面向量:

$$D = P(x,y)\frac{\partial}{\partial x} + Q(x,y)\frac{\partial}{\partial y} \tag{7.3}$$

一个非常数的函数, 如果它在 (7.3) 的解曲线上的取值恒为常数, 则称该函数为 (7.3) 的一个首次积分. 设 $f \in \mathbb{C}[x,y]$ 不可约. 我们称 $f=0$ 为 (7.3) 的不变代数曲线, 如果有 $Df = kf$, 这里 $k \in \mathbb{C}[x,y]$. 我们称 $m = \max\{\mathrm{tdeg}(P), \mathrm{tdeg}(Q)\}$ 为 (7.3) 的次数. 首次积分的研究可追溯到Poincaré 和 Darboux. 平面向量场的基本函数首次积分与平面向量场的不变代数曲线密切相关. Darboux 证明了平面多项式向量场的不变代数曲线的次数存在一个上界, 并且如果不变代数曲线的数目不小于 $m(m+1)/2+1$, 则该平面向量存在首次积分. Jouanolou 进一步证明了如果不变代数曲线的数目不小于 $m(m+1)/2+2$, 则该平面向量场存在有理函数首次积分. 更为一般的, Prelle 和 Singer([70])证明了如果一个平面向量场有基本函数的首次积分则该首次积分有特殊的形式, 并且它的积分因子由它的不变代数曲线完全确定. 这样, 确定平面向量场的基本函数的首次积分问题就转化为求它的不变代数曲线问题, 而这个问题的关键一步是求不变代数曲线的次数的上界. 这一问题现在一般被称为Poincaré问题或者 Darboux问题. Poincaré(Darboux)问题目前还没有完全解决. 现在所得到的结果基本都是在对平面向量场的奇点或者不变曲线的奇点做限制的条件下给出的, 参见[12].

2. 微分方程的吴特征列方法. 考虑具有任意变元个数与具有任意方程个数的非线性常微或偏微分多项式方程式组:

$$\mathbb{P} = \{p_1(x_1,\cdots,x_n)=0,\cdots,p_r(x_1,\cdots,x_n)=0\}$$

其中 $p_i \in \mathbb{K}\{\mathbb{X}\}, \mathbb{X}=(x_1,\cdots,x_n)$, $\mathbb{K}$ 是特征为零的域, 其微分算子是 $\delta_1,\cdots,\delta_m$. $\mathbb{K}$ 通常为 $\mathbb{Q}(t_1,\cdots,t_m)$. 此时微分算子是 $\dfrac{\partial}{\partial t_i}$. $\mathbb{P}=0$ 在一个 *微分泛域*$\mathbb{E}$ 上的所有解的集合称为由 $\mathbb{P}=0$ 定义的 *微分代数簇*, 记为 $\mathrm{Zero}(\mathbb{P})$.

这里采取的方法是从代数的观点来研究微分方程. 具体讲, 就是使用代数几何的工具研究微分方程解的维数、阶数、有理表示等问题. 因此, 这一领域也可以称为 *微分代数几何*. 下面介绍微分代数方程求解的吴消元法, 即微分情形的特征列方法.

升列与三角列是特征列方法的核心概念. 常微情形三角列的定义与代数情形相似. 一个微分多项式集合 $\mathcal{A} \subset \mathbb{K}\{\mathcal{X}\}$. 如果 $x_1, \cdots, x_n$ 可以重新排列为 $u_1, \cdots, u_q$, $y_1, \cdots, y_p$ $(p+q=n)$ 使得 $\mathcal{A}$ 中的微分多项式 $A_1, \cdots, A_p$ 可以被重排为如下形式, 则称 $\mathcal{A}$ 是一个 *三角列*.

$$\begin{aligned} A_1(\mathbb{U}, y_1) &= I_1 * y_{1,o_1}^{d_1} \ +\text{关于 } y_1 \text{ 的低阶、低次项} \\ A_2(\mathbb{U}, y_1, y_2) &= I_2 * y_{2,o_2}^{d_2} \ +\text{关于 } y_2 \text{ 的低阶、低次项} \\ &\cdots\cdots\cdots\cdots \\ A_p(\mathbb{U}, y_1, \cdots, y_p) &= I_p * y_{p,o_p}^{d_p} \ +\text{关于 } y_p \text{ 的低阶、低次项} \end{aligned} \tag{7.4}$$

其中 o_i 是 A_i 关于 y_i 的阶数, $y_{i,j}$ 是 y_i 的第 j 阶微分. 与代数情形相似, 给定 u_i 的一组值, 如果所有的初式 I_i 都不为零, 我们一般可以由 $A_1 = 0, A_2 = 0, \cdots, A_p = 0$ 顺序求解 $y_1, y_2, \cdots, y_p$. 换言之, $\mathrm{Zero}(AS/J)$ 可以被认为是一个确定的集合, 其中 $J = \prod_i I_i$.

偏微分多项式的升列不具有上面所写的严格三角列形式. 一般讲, 我们可以将其写为下面"三角化形式":

$$\begin{array}{l} A_{1,1}(\mathbb{U}, y_1), \cdots, A_{1,k_1}(\mathbb{U}, y_1) \\ A_{2,1}(\mathbb{U}, y_1, y_2), \cdots, A_{2,k_2}(\mathbb{U}, y_1, y_2) \\ \qquad\cdots\cdots\cdots\cdots \\ A_{p,1}(\mathbb{U}, y_1, \cdots, y_p), \cdots, A_{p,k_p}(\mathbb{U}, y_1, \cdots, y_p) \end{array} \tag{7.5}$$

在偏微情形, 我们还需要引进所谓 *可积条件* 使得相应三角列存在形式解. 满足可积条件的升列或三角列称为 *一致的*. 关于可积条件的定义见第四章.

与代数情形类似, 我们有

定理 7.1.2 *设 $\mathcal{A}$ 是任意一致微分升列, S 是 $\mathcal{A}$ 的初式与隔离子的乘积. 则有*

- *$\mathrm{Zero}(\mathcal{A}/S)$ 与 $\mathrm{Zero}(\mathrm{sat}(\mathcal{A}))$ 或者是空集或者是齐维的, 且其不可约分支的维数就是 $\mathcal{A}$ 的维数.*
- *对于一致不可约升列 $\mathcal{A}$, $\mathrm{Zero}(\mathcal{A}/S)$ 与 $\mathrm{Zero}(\mathrm{sat}(\mathcal{A}))$ 是不可约微分代数簇, 且其维数就是 $\mathcal{A}$ 的维数. 在常微情形, $\mathrm{Zero}(\mathrm{sat}(\mathcal{A}))$ 关于 $\mathbb{U}$ 的阶数即 $\mathcal{A}$ 的阶数: $\sum_{i=1}^{o} o_i$.*

微分情形下的吴特征列方法可以将一般形式的微分方程组 $\mathbb{P}$ 的解变为一致升列的解.

定理 7.1.3 (吴零点分解定理) 对于给定的微分多项式集合 $\mathbb{P}$, 存在算法确定一个有限的升列集合 (或三角列集合) $\mathcal{A}_j$, 使得

$$\mathrm{Zero}(\mathbb{P}) = \cup_j \mathrm{Zero}(\mathcal{A}_j/J_j) = \cup_j \mathrm{Zero}(\mathrm{sat}(\mathcal{A}_j)) \tag{7.6}$$

其中 J_j 是对应的 $\mathcal{A}_j$ 的初式与隔离子的乘积.

MMP 的函数 dwsolve 实现了上述零点分解. 通过零点分解, 有的微分方程组可以得到简化, 使其解变得一目了然. 下面举例说明.

例 7.1.4 微分方程 $g = u_t + 6\,u\,u_x + u_{xxx} - u_{xx} = 0$ 是KdV-Burgers方程 $u_t + 6\,u\,u_x - \alpha\,u_{xx} + \beta\,u_{xxx} = 0$ 的特殊形式。当 $\alpha = 0, \beta \neq 0$ 时, 为KdV方程, 当 $\alpha \neq 0, \beta = 0$ 时, 为Burgers方程. 方程 $g = 0$ 在变换 $(x, t, u) \longrightarrow (x + \epsilon\,\xi_1 + O(\epsilon^2), t + \epsilon\,\xi_2 + O(\epsilon^2), u + \epsilon\,\eta_1 + O(\epsilon^2))$ 下的无穷小Lie对称群确定方程组是

$$\begin{aligned}
&\frac{\partial\,\xi_1}{\partial u} = 0, \quad \frac{\partial\,\xi_2}{\partial x} = 0, \quad \frac{\partial\,\xi_2}{\partial u} = 0, \quad \frac{\partial^2\,\eta_1}{\partial u^2} = 0\\
&3\,\frac{\partial^2\,\eta_1}{\partial u \partial x} - 3\,\frac{\partial^2\,\xi_1}{\partial x^2} - \frac{\partial\,\xi_1}{\partial x} = 0, \quad -\frac{\partial\,\xi_2}{\partial t} + 3\,\frac{\partial\,\xi_1}{\partial x} = 0\\
&\frac{\partial^3\,\eta_1}{\partial x^3} - \frac{\partial^2\,\eta_1}{\partial x^2} + 6\,u\,\frac{\partial\,\eta_1}{\partial x} + \frac{\partial\,\eta_1}{\partial t} = 0\\
&-2\,\frac{\partial^2\,\eta_1}{\partial u \partial x} + 3\,\frac{\partial^3\,\eta_1}{\partial x^2 \partial u} - \frac{\partial^3\,\xi_1}{\partial x^3} - \frac{\partial\,\xi_1}{\partial t} + \frac{\partial^2\,\xi_1}{\partial x^2} + 6\,\eta_1 + 12\,u\,\frac{\partial\,\xi_1}{\partial x} = 0
\end{aligned} \tag{7.7}$$

用MMP可以得到下面的形式：

$$\begin{aligned}
&\frac{\partial \xi_1}{\partial x}, 6\eta_1 - \frac{\partial \xi_1}{\partial, t}, \frac{\partial \xi_1}{\partial u}\\
&\frac{\partial \xi_2}{\partial x}, \frac{\partial \xi_2}{\partial t}, \frac{\partial \xi_2}{\partial u}\\
&\frac{\partial \eta_1}{\partial x}, \frac{\partial \eta_1}{\partial t}, \frac{\partial \eta_1}{\partial u}
\end{aligned} \tag{7.8}$$

从上式中容易求解得 $\xi_1 = 6\,c_1\,t + c_2$, $\xi_2 = c_3$, $\eta_1 = c_1$.

求解过程如下：

```
>depend([xi1,xi2,eta1],[x,t,u]);
>ps:=[xi1[0,0,1],xi2[1,0,0],xi2[0,0,1],eta1[0,0,2],
3*eta1[1,0,1]-3*xi1[2,0,0]-xi1[1,0,0], -xi2[0,1,0]+
3*xi1[1,0,0],eta1[3,0,0]-eta1[2,0,0]+6*u*eta1[1,0,0]+
eta1[0,1,0],-2*eta1[1,0,1]+3*eta1[2,0,1]-xi1[3,0,0]-
xi1[0,1,0]+xi1[2,0,0]+6*eta1+12*u*xi1[1,0,0]];
```

```
>ord:=[xi1,xi2,eta1];
>dwsolve(ps,ord);
[[xi1[1,0,0],6*eta1-xi1[0,1,0],xi1[0,0,1],xi2[1,0,0],
xi2[0,1,0],xi2[0,0,1],eta1[1,0,0],eta1[0,1,0],eta1[0,0,1]]]
```

例 7.1.5 已知微分方程 $g = u_t + 6uu_x + u_{xxx} - u_{xx}$, 其中 $u = u(x,t)$. 经计算可知微分式 g 的确定方程组为

$$\begin{aligned}
&\frac{\partial \xi_1}{\partial u},\ \ \frac{\partial^2 \eta_1}{\partial u^2}, \frac{\partial \xi_2}{\partial x},\ \ \frac{\partial \xi_2}{\partial u}\\
&\frac{\partial \xi_2}{\partial t} - 3\frac{\partial \xi_1}{\partial x}, \frac{\partial^2 \eta_1}{\partial x \partial u} - \frac{\partial^2 \xi_1}{\partial x^2}\\
&\frac{\partial^3 \eta_1}{\partial x^3} + 6u\frac{\partial \eta_1}{\partial x} + \frac{\partial \eta_1}{\partial t}\\
&6\eta_1 + 3\frac{\partial^3 \eta_1}{\partial u \partial x^2} - \frac{\partial^3 \xi_1}{\partial x^3} - \frac{\partial \xi_1}{\partial t} - 6u\frac{\partial \xi_1}{\partial x} + 6u\frac{\partial \xi_2}{\partial t}
\end{aligned} \tag{7.9}$$

其中 $\xi_1 = \xi_1(u,x,t), \xi_2 = \xi_2(u,x,t), \eta_1 = \eta_1(u,x,t)$.

我们令 $\xi_1 < \xi_2 < \eta_1$ 且 $u < x < t$, 求得上述偏微分方程组的特征列为

$$\begin{aligned}
&\frac{\partial \xi_1}{\partial u},\ \ \frac{\partial^2 \xi_1}{\partial x^2},\ \ \frac{\partial^2 \xi_1}{\partial t^2},\ \ \frac{\partial^2 \xi_1}{\partial x \partial t}\\
&\frac{\partial \xi_2}{\partial u},\ \ \frac{\partial \xi_2}{\partial x},\ \ \frac{\partial \xi_2}{\partial t} - 3\frac{\partial \xi_1}{\partial x}\\
&6\eta_1 + 12\frac{\partial \eta_1}{\partial x} - \frac{\partial \eta_1}{\partial t}
\end{aligned} \tag{7.10}$$

可知方程 g 的对称群为

$$\eta_1 = 2C_1\ u + C_2, \xi_2 = -3C_2\ t + C_3, \xi_1 = 6C_2\ t - C_1\ x + C_4$$

求解过程如下:

```
>depend([xi1,xi2,eta1],[u,x,t]);
>ps:=[xi1[1,0,0],eta1[2,0,0],
      xi2[0,1,0],xi2[1,0,0],
      xi2[0,0,1]-3*xi1[0,1,0],
      eta1[1,1,0]-xi1[0,2,0],
      eta1[0,3,0]+6*u*eta1[0,1,0]+eta1[0,0,1],
      6*eta1+3*eta1[1,2,0]-xi1[0,3,0]-xi1[0,0,1]
      -6*u*xi1[0,1,0]+6*u*xi2[0,0,1]];
```

```
>ord:=[eta1,xi2,xi1];
>dwsolve(ps,ord);
[[6*eta1+12*xi1[0,1,0]*u-xi1[0,0,1],xi2[1,0,0],xi2[0,1,0],
xi2[0,0,1]-3*xi1[0,1,0],xi1[0,2,0],xi1[0,1,1],xi1[0,0,2],
xi1[1,0,0]]];
```

下面简要介绍吴零点分解定理在常微情形微分方程求解中的若干应用.

3. 微分预解式. 在 § 6.2 中, 我们证明可以通过预解式的计算将任意一个不可约代数簇双有理变换到一个不可约超曲面. 这一理论可以直接推广到常微分情形. 具体我们有

定理 7.1.6 设 $\mathcal{A}$ 是一个形如 (7.4) 的关于常微分多项式的不可约升列. 我们可以求得 $\mathbb{K}$ 中的元素 $c_1,\cdots,c_p$ 与新变元 $w=c_1y_1+\cdots+c_py_p$, 使得在变元顺序 $u_i<w<y_1<\cdots<y_p$ 下有

$$\mathrm{Zero}(\mathrm{sat}(\mathcal{A})\cup\{w-c_1y_1-\cdots-c_py_p\}/J)=\mathrm{Zero}(\mathrm{sat}(\mathcal{A}')/J)$$

其中 J 是 $\mathcal{A}$ 的初式与隔离子的乘积. $\mathcal{A}'$ 具有下面形式：

$$\begin{array}{c}
R(u_1,\cdots,u_q,w)\\
I_1(u_1,\cdots,u_q,w)y_1-U_1(u_1,\cdots,u_q,w)\\
I_2(u_1,\cdots,u_q,w)y_2-U_2(u_1,\cdots,u_q,w)\\
\cdots\cdots\cdots\cdots\\
I_p(u_1,\cdots,u_q,w)y_p-U_p(u_1,\cdots,u_q,w)
\end{array}$$

这里 R,I_i,U_i 均为变量 u 与 w 的微分多项式. 我们称 R 是素理想 $\mathrm{sat}(\mathcal{A})$ 的预解式. 由此得到不可约微分代数簇的预解式表示：

$$R(u_1,\cdots,u_q,w),y_1=\frac{U_1}{I_1},y_2=\frac{U_2}{I_2},\cdots,y_p=\frac{U_p}{I_p}$$

关于 c_i 的选取, 我们有下面结论：随机选取 $\mathbb{K}$ 中的元素 c_i, 则定理 7.1.6 成立的概率为 1.

4. 含参数微分方程的求解. 设$\mathbb{B}=\mathbb{K}\{u_1,\cdots,u_m\}$, $\mathbb{A}=\mathbb{B}\{x_1,\cdots,x_n\}$. 对于 $\mathbb{K}\{\mathcal{U},\mathcal{X}\}$ 中的微分多项式集合 $\mathbb{P}=\{p_1,\cdots,p_t\}$ 与 $\mathbb{D}=\{d_1,\cdots,d_r\}$, 考虑下面含参数的微分方程：

$$p_1=0\wedge\cdots\wedge p_t=0\wedge d_1\neq0\wedge\cdots\wedge d_r\neq0 \tag{7.11}$$

或等价地, 考虑 $\mathrm{Zero}(\mathbb{P}/\mathbb{D})$. 对于上述微分参数方程, 我们同样可以提出以下基本问题：

- 对于哪些参数值, 多项式方程 $\mathbb{P}=0, \mathbb{D}\neq 0$ 存在解?
- 怎样表示这些解?

与代数情形相似, 引入下列定义.

方程组 (7.11) 的解函数是一对 $(S,\mathcal{A})$, 其中 S 是 $\mathbb{E}^m$ 中一个拟代数簇, $\mathcal{A}$ 是 $\mathbb{B}\{\mathbb{X}\}-\mathbb{B}$ 中的一个三角列, 满足 (a) 对每一个 $u'\in S$, 记 $\mathcal{A}'$ 是将 u 替换为 u' 的结果, 则 $\mathrm{Zero}(\mathcal{A}'/J')$ (其中 J' 是将 $\mathcal{A}$ 中多项式的初式中的 u 替换为 u' 的结果) 是一个 $\dim(\mathcal{A})$ 维拟代数簇; (b) 对每一个 $x'\in\mathrm{Zero}(\mathcal{A}'/J')$, $(u',x')\in\mathrm{Zero}(\mathbb{P}/\mathbb{D})$. 我们称 (u',x') 是 $(S,\mathcal{A})$ 的一个解.

方程组 (7.11) 的一个 *覆盖* 是 (7.11) 的解函数集 $C=\{(S_1,\mathcal{A}_1),\cdots,(S_s,\mathcal{A}_s)\}$, 满足：每一个 $(u',x')\in\mathrm{Zero}(\mathbb{P}/\mathbb{D})$ 均为某一个 $(S_i,\mathcal{A}_i)$ 的解函数.

我们给出求参数系统 (7.11) 的覆盖的一个算法. 这一算法主要有三步：

- 调用吴零点分解定理将一般形式的方程组分解为三角形式.
- 对三角列进行投影运算可以转化为对三角列中的单个微分多项式进行投影运算. 而且, 关于参数的三角列不会被上述运算破坏.
- 求得各个三角列的投影后, 就可以很容易地给出相应的解函数.

设 $C=\{(S_1,\mathcal{A}_1),\cdots,(S_s,\mathcal{A}_s)\}$ 为 (7.11) 的覆盖. 我们有

$\dim(\mathcal{A}_i)$, $i=1,\cdots,s$ 是参数系统 (7.11) 所有可能的维数.

令 $M\subset C$ 为 m 维的 S_i, 则在 $\mathbb{K}(\mathbb{U})[\mathbb{X}]$ 中, 我们有

$$\mathrm{Zero}(\mathbb{P}/\mathbb{D})=\cup_{(S_j,\mathcal{A}_j)\in M}\mathrm{Zero}(\mathcal{A}_j/J_j\cup\mathbb{D})$$

例 7.1.7　考虑关于输入控制变量 u, 隐含变量 x 和输出变量 y 的控制系统

$$x'=ux^2+u^2x,\qquad y=x^2 \tag{7.12}$$

我们需要消去 x. 利用求覆盖的算法, $\mathrm{Zero}(7.12)=\mathrm{Zero}(y'^2-4u^2yy'-4u^2y^3+4u^4y^2, 2uyx-y'+2u^2y/u(y'-2u^2y))\cup\mathrm{Zero}(u,y',x^2-y/x)\cup\mathrm{Zero}(y,x)$.(7.12) 的一个覆盖为 $\{(\mathrm{Zero}(y'^2-4u^2yy'-4u^2y^3+4u^4y^2/u(y'-2u^2y)); 2uyx-y'+2u^2y), (\mathrm{Zero}(u,y'/y); x^2-y), (\mathrm{Zero}(y); x)\}$. 由此我们得到三个关系式. 更进一步, 我们给出隐含变量 x 在每一个输入输出关系集上的值.

注意到在 $\mathrm{Zero}(u,y'/y)$ 上 $x=\sqrt{y}$ 不能用 y 的值线性表示. 用控制的语言讲, 此时系统是不可观测的. 而在其他情形, x 可以用其他变量线性表示, 因而此时系统是可观测的.

5. 微分有理参数方程的隐式化.　我们在 6.5.4节讨论了单有理代数簇的有理表示的隐式化问题. 一个自然的想法就是我们能否将曲线、曲面的有理参数化以及隐式化

的结果推广到微分代数几何中去. 例如, 一个微分代数簇的隐式表示与参数表示分别为

$$y'^2 - 4yz^2 = 0; \quad y = u^2, \quad z = u'$$

这里 u 为参数. 微分有理参数化 $y = u^2, z = u'$ 也可以看作微分方程 $y'^2 - 4yz^2 = 0$ 的通解. 我们下面说明怎样使用 MMP 计算微分有理参数方程的隐式化. 微分方程的有理参数表示问题, 目前还没有任何结果.

设 $P_1, \cdots, P_n, Q_1, \cdots, Q_n$ 为 $\mathbb{K}\{\mathbb{U}\}$ 中的微分多项式. 我们称

$$x_1 = \frac{P_1(\mathbb{U})}{Q_1(\mathbb{U})}, \cdots, x_n = \frac{P_n(\mathbb{U})}{Q_n(\mathbb{U})} \tag{7.13}$$

是一个*微分有理参数表示*. 假定不是所有的 $P_i(\mathbb{U})$ 和 $Q_i(\mathbb{U})$ 都在 $\mathbb{K}$ 中, 并且 $\gcd(P_i(\mathbb{U}), Q_i(\mathbb{U})) = 1$.

微分有理参数表示 (7.13) 的*隐式理想*定义为

$$\mathbf{ID}(P, Q) = \{P \in \mathbb{K}\{x_1, \cdots, x_n\} | P(P_1/Q_1, \cdots, P_n/Q_n) \equiv 0\}$$

称 $\mathrm{Zero}(\mathbf{ID}(P, Q))$ 为 (7.13) 的*隐式簇*. 设 $\mathcal{V}$ 是一个微分代数簇, 如果 $\mathcal{V}$ 是 (7.13) 的隐式簇, 则称 $\mathcal{V}$ 是*单有理*的, 而 (7.13) 是 $\mathcal{V}$ 的一个*微分有理参数表示*.

设 $\mathbb{P} = \{F_1 = P_1 - x_1Q_1, \cdots, F_n = P_n - x_nQ_n\}$, $\mathbb{D} = \{Q_1, \cdots, Q_n\}$. 因为 $\mathbb{P}$ 在秩 $u_1 < \cdots < u_m < x_1 < \cdots < x_n$ 下是一个升列, 可以定义 $\mathrm{sat}(\mathbb{P})$. 与代数情形相似, 可以如下计算隐式理想的特征列. 在秩 $x_1 < \cdots < x_n < u_1 < \cdots < u_m$ 下计算 $\mathrm{Zero}(\mathbb{P}/\mathbb{D})$ 的一个零点分解. 利用吴零点分解定理有

$$\mathrm{Zero}(\mathbb{P}/\mathbb{D}) = \cup_i \mathrm{Zero}(\mathrm{sat}(\mathcal{A}_i)/\mathbb{D})$$

在上面的分解中, 存在一个 $\mathcal{A}_k$, 使得其中的微分多项式以 P_i/Q_i 为零点. 可以证明 $\mathrm{sat}(\mathcal{A}_k) = \mathrm{sat}(\mathbb{P})$. 通过重新命名变量, 我们可以把 $\mathcal{A}_k$ 写成下面的形式：

$$\begin{aligned}
&A_1(x_1, \cdots, x_{d+1}), \cdots, A_{n-d}(x_1, \cdots, x_n),\\
&B_1(x_1, \cdots, x_n, t_1, \cdots, t_{s+1}), \cdots, B_{m-s}(x_1, \cdots, x_n, t_1, \cdots, t_m)
\end{aligned} \tag{7.14}$$

其中 $d + s = m$. 由上面讨论, 有以下结论：

定理 7.1.8　*有理参数方程* (7.13) *的隐式理想是* $\mathbf{ID}(P, Q) = \mathrm{sat}(A_1, \cdots, A_{n-d})$.

与代数情形不同, 此时我们只能求出隐式理想的特征列, 而不能求出其基.

例 7.1.9　*考虑下列 DRPEs*

$$x = u^2, y = u' \tag{7.15}$$

令 $\mathbb{P}=\{x-u^2, y-u'\}$. 在变量序 $x<y<u$ 下, 由吴零点分解定理, 我们有

$$\begin{aligned}\text{Zero}(\mathbb{P})=&\text{Zero}(\{4xy^2-x'^2, 2yu-x'\}/xy)\\ &\cup\text{Zero}(\{x', y, -u^2+x\}/u)\cup\text{Zero}(\{x,y,u\})\end{aligned}$$

在MMP中命令如下:

```
>depend([u,y,x],[t]);
>ps:=[x-u^2, y-u[1]];
>vs:=[u,y,x];
>dwsolve(ps,vs,[],"ritt");
```

输出结果为

$[[2yu-x[1], 4xy^2-x[1]^2], [-u^2+x, y, x[1]], [u, y, x]]$.

只有 $4xy^2-x'^2$ 在 $x=u^2, y=u'$ 时为零. 所以, 隐式理想为 $\text{sat}(4xy^2-x'^2)$.

如果 (7.13) 的隐式理想是 m 维的, 就称 (7.13) 的参数 $u_1,\cdots,u_m$ 是独立的. 我们有

定理 7.1.10 假设我们已经构造了 (7.14), 那么 $\mathbf{ID}(P,Q)$ 的维数是 $d=m-s>0$. 因此, 参数是独立的, 当且仅当 $s=0$.

如果 (7.13) 的参数不独立, 我们可以找到新的有理参数方程

$$x_1=\overline{P}_1/\overline{Q}_1,\cdots,x_n=\overline{P}_n/\overline{Q}_n$$

它和 (7.13) 有相同的隐式簇, 但它的参数是独立的. 算法与代数情形相似. 现举例说明.

例 7.1.11 考虑下列 DRPEs

$$x=u^2+2uv^2+v^4,\quad y=u'+2vv' \tag{7.16}$$

令 $\mathbb{P}=\{x-u^2-2uv^2-v^4, y-u'-2vv'\}$. 在变量序 $y<x<v<u$ 下, 由吴零点分解定理 7.1.3, 我们有

$$\begin{aligned}\text{Zero}(\mathbb{P})=&\text{Zero}(\{2yu+2v^2y-x', 4xy^2-x'^2\}/x'y)\\ &\cup\text{Zero}(\{-u^2-2v^2u+x-v^4, x', y\}/u-v^2)\\ &\cup\text{Zero}(\{u+v^2, x, y\})\end{aligned}$$

在MMP中命令如下:

```
>depend([u,v,x,y],[t]);
>psf:=[x - u^2-2*u*v^2-v^4, y - u[1]-2*v*v[1]];
>vs:=[u,v,x,y];
>dwsolve(ps,vs,[],"ritt");
```

输出结果为

$[[2yu+2v^2y-x[1], 4xy^2-x[1]^2], [-u^2-2v^2u+x-v^4, x[1], y], [u+v^2, x, y]]$.

可以验证

$$\{2yu+2v^2y-x', 4xy^2-x'^2\}$$

是对应 (7.14) 的升列. 由定理 7.1.10 知 u 和 v 不是独立的. 上述升列的初式与隔离子之积等于 $J=x'y$. 利用 x'^2-4xy^2 消去 J 中的 x', 我们有 $K=4xy^4$. 为求出一组独立的参数, 我们只需要选择 v 的一组值满足 $M=4xy^4=4(u^2+2uv^2+v^4)(u'+2vv')^4$ 不为零. 选取 $v=0$. 方程 (7.16) 变为具有独立参变量的方程 (7.15).

我们称有理方程 (7.13) 是正则的, 如果对于隐式簇的一个母点 $(a_1,\cdots,a_n)$, 只存在一个 $\tau=(\tau_1,\cdots,\tau_m)\in\mathbb{E}^m$, 使得 $a_i=P_i(\tau)/Q_i(\tau)$, $i=1,\cdots,n$.

假设 (7.13) 的参数 $u_1,\cdots,u_m$ 是独立的, 即 $s=0$. 那么 (7.14) 就变为

$$\begin{aligned}&A_1(x_1,\cdots,x_{m+1}),\cdots,A_{n-m}(x_1,\cdots,x_n),\\&B_1(x_1,\cdots,x_n,u_1),\cdots,B_m(x_1,\cdots,x_n,u_1,\cdots,u_m)\end{aligned}\tag{7.17}$$

我们有

定理 7.1.12 使用上面的符号. (7.13) 是正则的, 当且仅当 $B_i=I_iu_i-U_i$ 关于 u_i 是线性的. 在这种情况下, 逆映射是

$$u_1=U_1/I_1,\cdots,u_m=U_m/I_m$$

其中 I_i 和 U_i 是 $\mathbb{K}\{\mathbb{X}\}$ 中的多项式.

如果 $m=1$ 且 (7.13) 非正则, 我们可以找出新的参变量 $s=f(u_1)/g(u_1)$ 使得 (7.13) 关于 s 的重新参数化

$$x_1=\frac{F_1(s)}{G_1(s)},\cdots,\ x_n=\frac{F_n(s)}{G_n(s)}\tag{7.18}$$

是正则的. 这里 f 和 g 属于 $\mathbb{K}\{u_1\}$. 计算过程与代数情形相似. 用下面例子说明.

例 7.1.13 考虑下列 *DRPEs*

$$x=u^4, y=2uu'\tag{7.19}$$

令 $\mathbb{P}=\{x-u^4, y-2uu'\}$. 在变量序 $x<y<u$ 下, 由吴零点分解定理 7.1.3, 我们有

$$\begin{aligned}\mathrm{Zero}(\mathbb{P})=&\mathrm{Zero}(\{4xy^2-x'^2, 2yu^2-x'\}/xy)\\&\cup\mathrm{Zero}(\{x', y, -u^4+x\}/u)\cup\mathrm{Zero}(\{x,y,u\})\end{aligned}$$

在 MMP中命令如下：

```
>depend([u,y,x],[t]);
>ps:=[x-u^4, y-2*u*u[1]];
>vs:=[u,y,x];
>dwsolve(ps,vs,[],"ritt");
```

输出结果为

$[[2yu^2 - x[1], 4xy^2 - x[1]^2], [-u^4 + x, y, x[1]], [u, y, x]]$.

由于 $2yu^2-x'$ 关于 u 非线性, 由定理 7.1.12, (7.19) 非正则. $u=\sqrt{\frac{x'}{2y}}$ 是一个逆映射. 为求正则的微分有理方程, 将 (7.19) 代入下式：$\bar{B}_1 = 2yw^2 - x' = 4uu'w^2 - 4u^3u'$. 选择新的参变量 $s = (4u^3u')/(4uu') = u^2$. 消去 $s = u^2$ 和 (7.19) 中的 u, 得到新的参数方程

$$x = s^2, \quad y = s' \tag{7.20}$$

方程 (7.20) 是正则的且与 (7.19) 具有相同的隐式理想. 这实际上就是 (7.15). 这一正则参数方程的逆映射为 $s = \dfrac{x'}{2y}$, 在满足 $y \neq 0$ 的映像的点上有意义.

对于 $\mathbb{K}$ 的扩域 $\mathbb{E}$, 有理参数方程 (7.13) 在 $\mathbb{E}^n$ 中的映像是指

$$\mathbf{IM}(P,Q) = \{(\eta_1,\cdots,\eta_n) \in \mathbb{E}^n | \exists\tau \in \mathbb{E}^m (\eta_i = P_i(\tau)/Q_i(\tau))\}$$

显然

$$\begin{aligned}\mathbf{IM}(P,Q) &= \{(x_1,\cdots,x_n) | \exists\tau \in \mathbb{E}^m (Q_i(\tau)x_i - P_i(\tau) = 0 \wedge Q_i(\tau) \neq 0)\} \\ &= \mathrm{Proj}_{u_1,\cdots,u_m}\mathrm{Zero}(\mathbb{P}/\mathbb{D})\end{aligned}$$

利用微分投影定理 4.5.7, 我们可以得到 $\mathbb{K}\{\mathbb{U}\}$ 中的多项式集合 $\mathcal{S}_i$ 和多项式 d_i, $i = 1,\cdots,t$, 使得

$$\mathbf{IM}(P,Q) = \cup_{i=1}^t \mathrm{Zero}(\mathcal{S}_i/d_i)$$

例 7.1.14　例 7.1.9 的续. 计算参数方程 (7.15) 的映像. 利用投影定理, 我们有

$$\begin{aligned}\mathbf{IM}(P,Q) &= \mathrm{Zero}(\{4xy^2 - x'^2\}/xy) \cup \mathrm{Zero}(\{x',y\}/x) \cup \mathrm{Zero}(\{x,y\}) \\ &= \mathrm{Zero}(\{4xy^2 - x'^2\}/xy) \cup \mathrm{Zero}(\{x',y\})\end{aligned}$$

在MMP中命令如下：

```
>depend([u,y,x],[t]);
>ps:=[x-u^2, y-u[1]];
>vs:=[u,y,x];
>proj(ps,[],vs,[u],"diff");
```

输出结果为
$[[[4xy^2-x[1]^2],[x,y]],[[x[1],y],[x]],[[x,y],[1]]]$.
我们还可以得到下列典型表示：

$$\begin{aligned}\mathbf{IM}(P,Q)&=\mathrm{Zero}(\{4xy^2-x'^2\}/xy)\cup\mathrm{Zero}(\{x',y\})\\&=\mathrm{Zero}(\{4xy^2-x'^2\})-(\mathrm{Zero}(xy)-\mathrm{Zero}(\{x',y\}))\\&=\mathrm{Zero}(\{4xy^2-x'^2\})-\mathrm{Zero}(x/y)\end{aligned}$$

上面我们利用了在 $4xy^2-x'^2=0$ 条件下 $\mathrm{Zero}(y)=\mathrm{Zero}(x',y)$ 的性质.
我们看到, 在微分情形, 即使是多项式参数方程的映像也不是整个隐式代数簇.

§7.2　常微分方程的初等函数解

吴零点分解定理可以将一般形式的微分方程求解变为单个方程的求解. 本章考虑一类简单的单个方程：有理系数的常微分方程 $F(y)=0$, 其中 $F(y)\in\mathbf{Q}(x)\{y\}$. 在本节, 我们记 $y_i=\frac{d^iy}{d^ix}$.

1. 初等函数与 Liouvillian 函数. 首先解释初等函数的明确意义. 由于我们考虑的是常微分方程, 只需注意单个变量 x 的初等函数. 又由于这些函数是微分方程的解, 其中不可避免将包含任意常数. 为此, 引进下面概念：

泛常数域$\widetilde{\mathbf{Q}}$ 是指在有理数域 $\mathbf{Q}$ 中添加了任意多个任意常数后再做代数闭包所得到的代数闭域. 可以证明, 微分方程的解的系数总可以在 $\widetilde{\mathbf{Q}}$ 中取到.

最简单的函数是 *多项式函数*, 即由变量 x 与泛常数域$\widetilde{\mathbf{Q}}$ 中的有限个元素经过有限次加、减、乘法运算所得到的表达式. 我们总可以将多项式函数写为如下形式：

$$\hat{y}=a_mx^m+a_{m-1}x^{m-1}+\cdots+a_0\tag{7.21}$$

其中 $a_i\in\widetilde{\mathbf{Q}}$. 如果允许除法运算, 则得到 *有理函数*：

$$\hat{y}=\frac{a_mx^m+a_{m-1}x^{m-1}+\cdots+a_0}{b_nx^n+b_{n-1}x^{n-1}+\cdots+n_0}\tag{7.22}$$

其中 $a_i,b_j\in\widetilde{\mathbf{Q}}$. $\deg(\hat{y})=\max\{m,n\}$ 称为有理函数的次数. 我们有时也说 $\hat{y}$ 的次数为 (m,n). 如果进一步允许代数运算, 则得到 *代数函数*：

$$G(x,\hat{y})=\sum_{j=0}^{n}\sum_{i=0}^{m_j}a_{i,j}x^i\hat{y}^j=0\tag{7.23}$$

这里 $a_{i,j} \in \widetilde{\mathbf{Q}}$ 并且 $G(x,y)$ 在 $\widetilde{\mathbf{Q}}[x,y]$ 上不可约. x 的代数函数 $\hat{y}$ 是 $G(x,\hat{y})=0$ 的解. 很明显, 多项式函数与有理函数是代数函数的特例.

例 7.2.1　*考虑下面 x 的多项式函数：*

$$\hat{y} = cx + u$$

其中 c 是任意常数, u 满足 $P(c,u)=0$ 而且 $P \in \mathbf{Q}[c,u]$ 是一个不可约多项式. 做微分, 我们有 $\hat{y}_1 = c$. 消去 c, u, 我们得到 $\hat{y}$ 是下面一阶微分方程：

$$P(y_1, y - xy_1) = 0$$

这个例子说明引入泛常数域的必要性. 同时也说明, 即便是一次多项式函数, 其相应的定义微分方程, 也可能是很复杂的.

为了将三角函数, 指数函数等常见超越函数包括进我们考虑的初等函数, 下面引进 Liouvillian 函数.

一个域 $\mathcal{F} = \widetilde{\mathbf{Q}}(x)(\theta_1, \cdots, \theta_n)$ 称为域 $\widetilde{\mathbf{Q}}(x)$ 的 *基本扩张*, 如果满足

1. θ_i 在 $\widetilde{\mathbf{Q}}(x)(\theta_1, \cdots, \theta_{i-1})$ 上代数; 或者
2. 存在 $g \in \widetilde{\mathbf{Q}}(x)(\theta_1, \cdots, \theta_{i-1})$, 使得 $\theta_i'/\theta_i = g'$ (即 $\theta_i = e^g$); 或者
3. 存在 $g \in \widetilde{\mathbf{Q}}(x)(\theta_1, \cdots, \theta_{i-1})$, 使得 $\theta_i' = g'/g$ (即 $\theta_i = \log(g)$), 这里 $'$ 指关于 x 做微分.

一个域 $\mathcal{F} = \widetilde{\mathbf{Q}}(x)(\theta_1, \cdots, \theta_n)$ 称为域 $\widetilde{\mathbf{Q}}(x)$ 的 *Liouvillian扩张*, 如果满足

1. θ_i 在 $\widetilde{\mathbf{Q}}(x)(\theta_1, \cdots, \theta_{i-1})$ 上代数; 或者
2. $\theta_i'/\theta_i \in \widetilde{\mathbf{Q}}(x)(\theta_1, \cdots, \theta_{i-1})$; 或者
3. $\theta_i' \in \widetilde{\mathbf{Q}}(x)(\theta_1, \cdots, \theta_{i-1})$.

由定义可以看出, 基本扩张是一种特殊的 Liouvillian 扩张. 我们将基本扩张 (Liouvillian 扩张) 中的元素称为 *基本函数*(*Liouvillian函数*).

注释 7.2.2　*目前对于微分方程符号解的算法研究主要集中在线性微分方程或者一些特殊的非线性微分方程, 如Riccati方程及向量场. 主要结果介绍如下.*

形式最简单的微分方程是 $y' = f(x)$, 其求解问题即有限项积分问题. 这个问题在19 世纪被 Abel 和 Liouville 广泛的研究过. Liouville 在 1833 年给出了一个基本函数具有基本函数不定积分时所具有的形式. 从算法方面的考虑, 则是由 Risch 最先从代数的观点开始研究. 在 1969 年, Risch 给出了第一个完整的求解一类特殊的基本函数(有理函数以及有理函数与指数函数、对数函数的复合函数)的不定积分的算法 ([71]) 并且随后描述了一个求一般的基本函数有限项积分算法的基本思路. 此后有很多人对 [71] 中的算法做出了补充和改进, 并且在大部分的计算机代数系统中

实现了该算法. 对于代数函数的情形, Davenport 和 Trager 在引进一些新的思路并在改进 Risch 的方法后给出了一个算法 ([26]). 而后 Bronstein 推广了 Trager 所采用的技术, 给出求一般的基本函数积分的算法 ([7]). 在最坏的情形下, 这个算法的复杂度是指数的.

一般的线性齐次常微分方程是具有下面形式的方程:

$$L(y) = y^{(n)} + a_{n-1}y^{(n-1)} + \cdots + a_0 y = 0$$

这里 a_i 为基本函数或者 Liouvillian 函数. 众所周知, 齐次线性的常微分方程的所有解组成了一个线性空间. 对于线性常微分方程解析解在 19 世纪有过大量的研究. 线性方程求解的突破由 Kovacic 做出. 在 [57] 中, Kovacic 给出一个有效的算法判定一个二阶有理函数系数的线性齐次常微分方程是否具有 Liouvillian 函数解, 如果解存在则求出它的解空间的一组基. Kovacic 主要利用了二阶微分 Galois 群子群的分类理论将方程转化为求有理函数解或者代数函数解问题, 然后通过分析方程的奇点等给出算法. Kovacic 的算法非常清楚, 效率也很高. 对于一般的高阶线性常微分方程, Singer 建立了一般的框架 ([74]). Singer 的算法效率比较低. 一个自然的提高效率的方法就是将高阶常微分方程分解成低阶的常微分方程. 一般的, 线性常微分方程的 Liouvillian 函数解问题可以通过变换 $y = e^{\int z dx}$ 将其转化为求 Riccati 方程的代数函数解问题. 对于二阶以及三阶常微分方程的情形, Singer 与 Ulmer 利用群论中的技术给出了方程相应的 Riccati 方程的代数函数解的次数的最好的界, 从而得到有效的求 Liouville 函数解的算法 ([75]).

2. 微分方程的通解. 非线性微分方程的初等函数解的问题要困难的多. 实际上, 即使是对于下面两种最简单的情形都没有完整解答:

问题 1 求任意常微方程 $F(y) = 0$ 的多项式解.

问题 2 求一阶常微方程 $F(y) = 0$ 代数函数解.

我们将考虑一个相对简单的问题: 求微分方程的 *初等函数通解*. 令 $F \in \mathbb{K}\{y\} \setminus \mathbb{K}$ 为一个不可约的微分多项式并且令

$$\Sigma_F = \{A \in \mathbb{K}\{y\} | SF \equiv 0 \mod\{P\}\} \tag{7.24}$$

这里 $\{P\}$ 为由 P 生成的完全微分理想, S 为 P 的隔离子. 则 Σ_P 是一个素微分理想并且一个微分多项式 $Q \in \Sigma_P$ 当且仅当 $\mathrm{dprem}(Q, P) = 0$.

令 Σ 是 $\mathbb{K}\{y\}$ 中的非平凡的素微分理想, 即 $\Sigma \neq \{0\}$. 我们称 Σ 的零点 η 为 Σ 的 *母点*. 如果对于任意的微分多项式 Q: 若 $Q(\eta) = 0$, 则有 $Q \in \Sigma$. 我们知道一个 (微分) 理想是素的当且仅当它存在母点. 利用这一点, 我们可以给出下面 Ritt 所给的关于通解的严格定义:

对于一个不可约的微分多项式 F, 我们称(7.24)中所定义的 Σ_F 的任意一个母点为常微分方程 $F=0$ 的 *通解*. 对于一个不可约的微分多项式 F, 设 S 为 F 的隔离子, 则我们称常微分方程 $F=0$ 与 $S=0$ 的公共解为 $F=0$ 的 *奇异解*.

一个常微分方程 $F=0$ 的通解在通常意义下是指具有 o 个代数无关的参数的解, 这里 $o=\mathrm{ord}(F)$. 当这些参数被赋予某些值时就得到了 $F=0$ 的一个解. 这一定义与上述 Ritt 的定义是等价的.

接下来, 我们将用几个例子来说明上面所给出的定义.

例 7.2.3 *考虑最简单的Riccati方程：$y_1+y^2=0$. 容易验证：$y=\dfrac{1}{x+c}$ 是它的有理函数通解.*

例 7.2.4 *考虑常微分方程：$xy_1-ny=0$. 容易验证：$y=cx^n$ 是它的多项式通解. 从这个简单的例子, 我们可以看出通解的次数不仅与原常微分方程的次数有关, 而且还与常微分方程的常数系数有关系.*

例 7.2.5 *考虑常微分方程：*

$$F=8y^3y_1^3+12x^8y^2y_1^2-36x^7y^3y_1+27x^6y^4+27x^6=0$$

令 $G(x,y)=y^2+cx^3+\sqrt{c^3-1}$, 这里 c 为任意常数. 容易知道, $G(x,y)$ 为 $\overline{\mathbf{Q}}(c,\sqrt{c^3-1})$ 上不可约的多项式. 经计算可知 $\mathrm{dprem}(F,G)=0$, 由通解的通常意义下的定义可知 $G(x,y)=y^2+cx^3+\sqrt{c^3-1}=0$ 是 $F=0$ 的代数函数通解.

关于代数函数通解, 我们有下述结果：

引理 7.2.6 *令 $G(y)\in\mathbb{Q}(x)[y]$ 并且在 $\widetilde{\mathbf{Q}}(x)[y]$ 上不可约. 如果 $G(y)=0$ 的一个解是常微分方程 $F=0$ 的一个解, 则 $G(y)=0$ 的每个解都是 $F=0$ 的解.*

因为 $G(y)$ 在 $\overline{\mathbf{Q}}(x)[y]$ 上不可约, 所以 $G(y)=0$ 的每个解都是它的一般零点. 所以 $\mathrm{dprem}(F,G)=0$, 也就是说存在整数 k,l 及微分多项式 P,Q, 使得

$$S^kI^lF=PG'+QG$$

这里 S,I 分别为 G 的隔离子和初式, G' 为 G 关于 x 的一次微分. 因为 $G(y)=0$ 的每个解都是一般零点, 所以它不是 $S=0$ 或者 $I=0$ 的解, 所以 $G(y)=0$ 的每个解都是 $F=0$ 的解.

3. 代数函数通解存在的充要条件. 我们知道 $y_n=0$ 的所有解为任意次数不超过

n 的多项式. 我们下面将这个结论推广到任意代数函数. 令

$$\mathcal{M}_{k,l}(y)=\begin{pmatrix} \binom{k+1}{0}y_{k+1} & \binom{k+1}{1}y_k & \cdots & \binom{k+1}{l}y_{k+1-l} \\ \binom{k+2}{0}y_{k+2} & \binom{k+2}{1}y_{k+1} & \cdots & \binom{k+2}{l}y_{k+2-l} \\ \vdots & \vdots & & \vdots \\ \binom{k+l+1}{0}y_{k+l+1} & \binom{k+l+1}{1}y_{k+l} & \cdots & \binom{k+l+1}{l}y_{k+1} \end{pmatrix}$$

$\mathbb{D}_{k,l}=\det(\mathcal{M}_{k,l}(y))$, 这里 $\binom{k}{i}$ 表示二项式系数并且当 $i>k$ 时, $\binom{k}{i}=0$. 我们有

定理 7.2.7 *假设 $\bar{y}$ 为 $\mathbb{K}$ 的泛扩域中的元素. 那么 $\mathbb{D}_{k,l}(\bar{y})=0$ 当且仅当 $\bar{y}$ 为分子次数不超过 k、分母次数不超过 l 的有理函数. 因此一个微分方程 $F(y)=0$ 的通解是有理函数的充要条件是存在非负整数 n,m 使得 $\mathrm{dprem}(\mathbb{D}_{m,n}(y),F)=0$.*

```
>depend([y], [t]);
>F1:=diff(y, t)^2-2*y;
>dprem(diff(y, t, 3), F1, y);
```

以上计算说明 $y'^2-2y=0$ 的通解是次数为 2 的多项式.

```
>depend([y], [t]);
>F2:=diff(y, t)+y^2;
>dprem(DRAT(y, 0, 1), F2, y);
```

这里 $\mathrm{DRAT}(y,k,l)=\mathbb{D}_{k,l}$. 以上计算说明 $y'+y^2=0$ 的通解是次数为 $(0,1)$ 的有理函数.

为了求代数函数解, 对于任意的非负整数 h,α,k, 令

$$\mathcal{M}_{(h,\alpha;k)}(y)=\begin{pmatrix} \binom{k+1}{0}y_{k+1} & \binom{k+1}{1}y_k & \cdots & \binom{k+1}{\alpha}y_{k+1-\alpha} \\ \binom{k+2}{0}y_{k+2} & \binom{k+2}{1}y_{k+1} & \cdots & \binom{k+2}{\alpha}y_{k+2-\alpha} \\ \vdots & \vdots & & \vdots \\ \binom{k+h+1}{0}y_{k+h+1} & \binom{k+h+1}{1}y_{k+h} & \cdots & \binom{k+h+1}{\alpha}y_{k+h+1-\alpha} \end{pmatrix}$$

为 $(h+1)\times(\alpha+1)$ 阶矩阵. 令 $\underline{\alpha}=(\alpha_1,\alpha_2,\cdots,\alpha_n)\in\mathbb{Z}_{\geqslant 0}^n$, $\alpha_0\in\mathbb{Z}_{\geqslant 0}$, 这里 $\mathbb{Z}_{\geqslant 0}$ 表示非负整数的集合. 令

$$\mathcal{M}_{(\alpha_0;\underline{\alpha})}(y)=(\mathcal{M}_{(h,\alpha_1;\alpha_0)}(y)|\mathcal{M}_{(h,\alpha_2;\alpha_0)}(y^2)|\cdots|\mathcal{M}_{(h,\alpha_n;\alpha_0)}(y^n))$$

为 $(h+1)\times(h+1)$ 阶方阵, 这里 $h+1=\alpha_1+\alpha_2+\cdots+\alpha_n+n$. 再令

$$\mathbb{D}_{(\alpha_0;\underline{\alpha})}=\det(\mathcal{M}_{(\alpha_0;\underline{\alpha})}(y))$$

这就是我们所要构造的微分多项式. 下面我们来看看 $\mathbb{D}_{(\alpha_0;\underline{\alpha})}$ 的一些特殊形式.

例 7.2.8　当 $n=1, \alpha_1=0$ 时, $\mathbb{D}_{(\alpha_0;0)}$ 就是 y_{α_0}. 当 $n=1$ 时, $\mathbb{D}_{(\alpha_0;\alpha_1)}$ 就是本节开头所定义的 $\mathbb{D}_{\alpha_0,\alpha_1}$. 当 $n=2, \alpha_1=\alpha_2=1, \alpha_0=2$ 时, 有 $h=4$.

$$\mathbb{D}_{(2;(1,1))} = \begin{pmatrix} y_3 & 3y_2 & (y^2)_3 & 3(y^2)_2 \\ y_4 & 4y_3 & (y^2)_4 & 4(y^2)_3 \\ y_5 & 5y_4 & (y^2)_5 & 5(y^2)_4 \\ y_6 & 6y_5 & (y^2)_6 & 6(y^2)_5 \end{pmatrix}$$

这里 $(y^2)_k$ 是指 y^2 关于 x 的 k 次微分.

定理 7.2.9　$\widetilde{\mathbf{Q}}(x)$ 中的元素 $\bar{y}$ 是 $\mathbb{D}_{(\alpha_0;\underline{\alpha})}=0$ 的解, 当且仅当 $\bar{y}$ 为满足方程(7.23)的代数函数, 并且 $m_j \leqslant \alpha_j,\ j=0,1,\cdots,n$. 因此一个微分方程 $F(y)=0$ 的通解是代数函数的充要条件是存在非负整数 n 与非负整数列 $\alpha_0=(n_1,\cdots,n_k)$, 使得 $\mathrm{prem}(\mathbb{D}_{(\alpha_0;n)},F)=0$.

定理 7.2.9 所给出的充要条件事实上已经是一种可以计算的条件, 但是要利用它来判定一个常微分方程是否有代数函数通解需要确定代数函数通解的次数. 一旦知道代数函数通解的次数, 我们就可以利用吴零点分解定理将通解求出来. 下面说明, 对于某种特殊的微分方程, 代数函数通解的次数是可以确定的.

4. 一阶自治常微分方程的有理函数通解.　我们知道, 只要确定了微分方程的代数函数通解或有理函数通解的次数的上界, 就可以进行求解. 对于一阶自治 (即常系数) 常微分方程, 我们可以给出它的代数函数通解与有理函数通解次数的上界. 而且, 我们还可以给出计算其代数函数解的多项式时间算法.

我们先来看看一阶自治常微分方程通解的结构. 对于自治的常微分方程以及常数 c, 在对变元 x 做平移 $x+c$ 后, 它的解集是不变的. 设 $\overline{\mathbf{Q}}$ 是 $\mathbf{Q}$ 的代数闭包. 我们有：如果 $G(x,y)\in\overline{\mathbf{Q}}[x,y]$ 不可约, 且 $G(x,y)=0$ 为 $F=0$ 的代数函数解, 那么 $G(x+c,y)=0$ 也是 $F=0$ 的代数函数通解, 这里 c 为任意常数. 所以一阶自治常微分方程通解具有形式 $G(x+c,y)=0$.

相似地, 令

$$\bar{y}=\frac{\bar{a}_n x^n+\bar{a}_{n-1}x^{n-1}+\cdots+\bar{a}_0}{x^m+\bar{b}_{m-1}x^{m-1}+\cdots+\bar{b}_0}\in\overline{\mathbf{Q}}(x)$$

为 $F=0$ 的一个非平凡的有理函数解. 那么

$$\hat{y}=\frac{\bar{a}_n(x+c)^n+\cdots+\bar{a}_0}{(x+c)^m+\cdots+\bar{b}_0}$$

是 $F=0$ 的有理函数通解, 这里 c 是任意常数. 因此, 我们只需要求 $F=0$ 的非平凡有理函数或代数函数解, 就可以得到其通解.

对于一阶自治常微分方程的有理函数与代数函数解, 我们有以下结果.

定理 7.2.10 若 $F=0$ 有有理函数通解 $\hat{y}$, 那么我们有

$$\deg(\hat{y})=\deg(F,y_1)$$
$$\deg(F,y_1)-1\leqslant\deg(F,y)\leqslant 2\deg(F,y_1)$$

上述定理确定了有理函数通解的次数, 同时还给出了具有有理函数通解的常微分方程 $F(y,y_1)=0$ 中 y 的次数与 y_1 的次数所要满足的关系. 由此, 我们得到下面判定一个一阶自治常微分方程具有有理函数通解的判定定理.

定理 7.2.11 令 $d=\deg(F,y_1)$. 那么 $F=0$ 有有理函数通解当且仅当微分伪余式 $\mathrm{dprem}(\mathbb{D}_{d,d},F)=0$.

定理 7.2.12 令 $G(x,y)\in\overline{\mathbf{Q}}[x,y]$ 不可约并且 $G(x,y)=0$ 为 $F=0$ 的代数函数解. 那么我们有

$$\deg(G,x)=\deg(F,y_1)$$
$$\deg(G,y)\leqslant\deg(F,y)+\deg(F,y_1)$$

对于一阶自治常微分方程的有理函数解, 我们还可以进一步给出多项式时间算法. 我们将用 Padé 逼近构造有理函数解. Padé 逼近通过给定函数在零点处的 Taylor 级数展开来构造所求的有理函数. 它的定义如下：对于一个形式幂级数 $A(x)=\sum_0^\infty a_jx^j$ 以及两个非负整数 L 和 M, $A(x)$ 的 (L,M) Padé 逼近是指下面的有理分式：

$$[L\setminus M]=\frac{P_L(x)}{Q_M(x)}$$

满足

$$A(x)-\frac{P_L(x)}{Q_M(x)}=O(x^{L+M+1})$$

这里 $P_L(x)$ 为一个次数不超过 L 的多项式, $Q_M(x)$ 为一个次数不超过 M 的多项式. 而且 $P_L(x)$ 与 $Q_M(x)$ 互素, $Q_M(0)=1$.

令 $P_L(x)=\sum_0^L p_ix^i$, $Q_M(x)=\sum_0^M q_ix^i$. 我们可以通过解关于变元 p_i 与 q_i 的线性方程组来求 $P_L(x)$ 和 $Q_M(x)$：

$$\begin{gathered}
a_0=p_0\\
a_1+a_0q_1=p_1\\
\cdots\cdots\cdots\cdots\\
a_L+a_{L-1}q_1+\cdots+a_0q_L=p_L\\
a_{L+1}+a_Lq_1+\cdots+a_{L-M+1}q_M=0\\
\cdots\cdots\cdots\cdots\\
a_{L+M}+a_{L+M-1}q_1+\cdots+a_Lq_M=0
\end{gathered}\tag{7.25}$$

这里如果 $i < 0$, 则 $a_i = 0$ 以及如果 $j > M$, 则 $q_j = 0$. 我们可以将它写成如下的矩阵形式：

$$\begin{pmatrix} a_{L-M+1} & a_{L-M+2} & \cdots & a_L \\ a_{L-M+2} & a_{L-M+3} & \cdots & a_{L+1} \\ \vdots & \vdots & & \vdots \\ a_L & a_{L+1} & \cdots & a_{L+M-1} \end{pmatrix} \begin{pmatrix} q_M \\ q_{M-1} \\ \vdots \\ q_1 \end{pmatrix} = -\begin{pmatrix} a_{L+1} \\ a_{L+2} \\ \vdots \\ a_{L+M} \end{pmatrix} \tag{7.26}$$

Padé 逼近有下面两个基本的性质：

定理 7.2.13 (Frobenius, Padé)　*如果 Padé 逼近存在, 则 Padé 逼近 $[L\setminus M]$ 唯一.*

定理 7.2.14 (Padé)　*$f(x)$ 为具有下面形式的有理函数：*

$$f(x) = \frac{p_l x^l + p_{l-1}x^{l-1} + \cdots + p_0}{q_m x^m + q_{m-1}x^{m-1} + \cdots + 1}$$

当且仅当它的 Padé 逼近 $[L\setminus M] = f(x)$, 对于所有的 $L \geqslant l$ 并且 $M \geqslant m$ 成立.

求有理函数解算法的基本步骤如下：

1. 令 $d = \deg(F, y_1)$. 计算 $F = 0$ 在 $x = \infty$ 处的 Laurent 级数解的前 $2d+1$ 项：$\bar{y}(x) = \sum_{i=-N}^{M} c_i x^i$. 如果该算法返回级数 $\bar{y}(x)$, 那么转入下一步. 否则, $F = 0$ 没有有理函数解, 算法终止.
2. 选择一整数 k 使得 $z(t) = t^k \bar{y}(\frac{1}{t})$ 是含有非零常数项的多项式.
3. 求 $z(t)$ 的 $2d$ 次 Padé 逼近 $r(t)$. 如果没有解, 那么 $F = 0$ 没有有理函数解, 算法终止.
4. 令 $\bar{y}(x) = x^k r\left(\frac{1}{x}\right)$. 将 $\bar{y}(x)$ 代入 F. 如果 $F(\bar{y})$ 恒等于零, 那么返回通解 $\hat{y}(x) = (x+c)^k r\left(\frac{1}{x+c}\right)$. 否则, $F = 0$ 没有有理函数通解, 算法终止.

在第一步, 可以在 $F(y, y_1) = 0$ 定义的代数曲线上任意选一点, 直接计算 Taylor 展开. 这样做一般需要在代数扩域上进行计算. 而上面算法则只包含 $\mathbf{Q}$ 中的运算, 求出的结果也在 $\mathbf{Q}(x)$ 中.

5. MMP/DiffEquation: 微分方程求解模块.　MMP 微分方程求解的主要功能均在应用模块 MMP/DiffEquation 中实现. 与前面提到的 MMP 函数不同, MMP/DiffEquation 是用 MMP 的编程语言实现的. 其源程序可以在 MMP 的网站上下载. MMP/DiffEquation 目前主要包括以下功能：

de_poly：求一阶自治常微分方程的多项式通解.

de_rat：求一阶自治常微分方程的有理函数通解.

de_taylor：求常微分方程的 Taylor 形幂级数解.

de_twa, de_twb： 求非线性偏微分方程的行波解.

我们已经在 § 3.5 介绍了模块 de_poly 的源程序. 在本节介绍了模块 de_rat 的算法. 其他模块将在本章余下的几节介绍. 下面看几个例子.

求一阶常系数微分方程多项式解的函数是 polysolve(F), 其中 F 是要求的微分方程.

```
>read("de_poly.txt");
>F:=y[1]^3-27*(y[0]-1)^2:
>polysolve(F);
```

上面的命令可以求得微分方程 $y'^3-27(y-1)=0$ 的通解：$(x+c)^3+1$.

由于 polysolve 不是 MMP 的内部函数, 而是用 MMP 的编程语言实现的一个程序包. 为了使用这一功能, 用户需要将文件 de_poly.txt 复制到当前目录下面, 然后用 read(“de_poly.txt”) 命令将其读入 MMP 中.

求一阶常系数微分方程有理函数解的函数是 ratsolve(F), 其中 F 是要求的微分方程. 考虑下面关于 x 的函数 $y(x)$ 的常微方程：

$$F=y'+y^2-2y+1=0$$

使用 ratsolve 可以得到 $F=0$ 的通解：$\hat{y}=\dfrac{x+1-c}{x}$, 其中 c 是任意常数.

```
>read("de_rat.txt");
>F:= y[1]+y[0]^2-2*y[0]+1:
>ratsolve(F);
```

§7.3　微分方程的形式幂级数解

一般的微分方程通常没有前面所说的基本函数解. 于是就需要考虑无穷级数解, 特别是幂级数解. 幂级数作为解析函数最普通的表示式, 不仅可以表示基本函数在某一邻域中的展开式, 而且还可以用来定义特殊函数, 比如超几何级数. 下面我们将介绍如何求解常微分方程 $F(y)=0$ 在 $x=0$ 处的形式幂级数解.

1. Taylor 幂级数解.　这里我们将沿用 § 7.2 中的记号. 假设 $F=0$ 是 n 阶常微分方程. $F^{(k)}$ 表示 F 关于变量 x 的 k 次微分, 则我们有

$$F^{(k)}=Sy_{n+k}+R$$

这里 $S=\partial F/\partial y_n$ 为 F 的隔离子, R 为阶数不超过 $n+k-1$ 的微分多项式. 给定初值：

$$y(0)=c_0, y_1(0)=c_1,\cdots,y_{n-1}(0)=c_{n-1}$$

如果存在常数 c_n 满足 $F(0,c_0,c_1,\cdots,c_{n-1},c_n)=0$ 并且 $S(0,c_0,c_1,\cdots,c_n)\neq 0$, 则可如下求 $F=0$ 在 $x=0$ 处的幂级数解的前 N 项：

1. 求 c_n 使得 $F(0,c_0,c_1,\cdots,c_{n-1},c_n)=0$ 并且 $S(0,c_0,c_1,\cdots,c_n)\neq 0$. 如果不存在满足条件的 c_n , 则算法终止.
2. $i:=n+1$ 以及 $\varphi(x):=c_0+c_1x+\dfrac{c_2x^2}{2}+\cdots+\dfrac{c_nx^n}{n!}$.
3. while $i\leqslant N$ do
   i. 将 $y=\varphi(x)$ 代入 $F(y)$.
   ii. $c:=F(x,\varphi(x),\varphi'(x),\cdots,\varphi^{(n)}(x))$ 中 x^{i-n} 的系数.
   iii. $\varphi(x):=\varphi(x)+\dfrac{(i-n)!*c*x^i}{i!*S(0,c_0,\cdots,c_n)}$.
   iv. $i:=i+1$.
4. 返回 $\varphi(x)$.

下面我们用两个例子来说明上面的方法.

例 7.3.1 *考虑微分方程*：

$$F=525-1230y+729y^2-15y_1^2-9y_1^3=0$$

由定理 7.2.10, 如果 $F=0$ 有多项式解 $\varphi(x)$, 则 $\deg(\varphi(x))=3$. 下面我们利用上面的方法求它的在给定初值 $y(0)=1$ 下的幂级数解的前 3 项.

1. 我们有 $F(0,1,y_1)=24-15y_1^2-9y_1^3$. 求的它的一个根 $y_1=1$. 此时我们有 $S(0,1,1)=-57\neq 0$.
2. $i=2$ 以及 $\varphi(x)=1+x$.
3. 将 $\varphi(x)$ 代入 F, 求得 $F(\varphi(x))=228x+729x^2$. 于是 $c=228$. 从而 $F=0$ 的幂级数解的前面 2 项为 $\varphi(x)=1+x+2x^2$.
4. 同样的, 我们求得 $F=0$ 的幂级数解的前面 3 项为 $\varphi(x)=1+x+2x^2+3x^3$.
5. 将 $\varphi(x)$ 代入 F, 得到 $F(\varphi(x))=0$. 所以 $\varphi(x)$ 为 $F=0$ 的多项式解.

求解过程如下：

```
>read("de_taylora.txt");
>read("de_taylorb.txt");
>F:=525-1230*y[0]+729*y[0]^2-15*y[1]^2-9*y[1]^3:
>taylorsolve(F,1,[1],10);
```

求 Taylor 幂级数解的函数是 taylorsolve(F,o,vs,n), 其中 F 表示待求解的微分方程，o 表示方程的阶数，vs 表示方程的解在零点的初值，n 表示需要展开的项.

例 7.3.2 *考虑微分方程*：

$$F=x^2+y_1y_2+xy_1+xy_2^2+x^4+y_3^4=0$$

以及给定初值：$y(0)=1, y_1(0)=2, y_2(0)=3$. 我们求它的幂级数解的前面 4 项.

1. 我们有 $F(0,1,2,3)=y_3^4+6$. 求的它的一个根 $y_3=Z$, 这里 Z 满足 $Z^4+6=0$. 此时我们有 $S(0,1,2,3)=4Z^3\neq 0$.
2. $i=4$ 以及 $\varphi(x)=1+2x+\dfrac{3x^2}{2}+\dfrac{Zx^3}{6}$.
3. 将 $\varphi(x)$ 代入 F, 求得
$$F(\varphi(x))=(2Z+20)x+4x^2+\frac{21Zx^2}{2}+\frac{3Z^2x^3}{2}+\frac{Zx^3}{2}+x^4$$
于是 $c=2Z+20$. 从而 $F=0$ 的幂级数解的第 4 项为 $-\dfrac{(2Z+20)x^4}{4!(4Z^3)}=\dfrac{(Z^2+10Z)x^4}{288}$.
4. 从而 $F=0$ 的幂级数解的前面 4 项为
$$\varphi(x)=1+2x+\frac{3x^2}{2}+\frac{Zx^3}{6}+\frac{(Z^2+10Z)x^4}{288}$$

求解过程如下：

```
>read("de_taylora.txt");
>read("de_taylorb.txt");
>F:=x^2+y[1]*y[2]+x*y[1]+x*y[2]^2+x^4+y[3]^4:
>taylorsolve(F,1,[1,1],4);
```

2. Puiseux 幂级数解. 当 $S(0,c_0,\cdots,c_n)=0$ 时, 无法利用上面的方法. 此时我们将用 Newton 多边形方法. Newton 多边形方法最初由 Newton 以及 Puiseux 在求解代数方程时提出. Briot 以及 Bouquet[8] 利用这一方法研究一阶常微分方程的奇异点. Fine 在 [31] 中给出了利用 Newton 多边形求解常微分方程的一个完整的描述. 此后 Grigoriev, Singer, Cano 以及 Hubert 等人将这一方法推广为求解一般的常微分方程以及偏微分方程和方程组 ([3], [11] 及 [52]). 我们将简单介绍一下 Newton 多边形方法, 具体的可参见上面所述的参考文献.

将 F 写成下面的形式：
$$F=\sum a_{\alpha,\beta_0,\cdots,\beta_n}x^\alpha y_0^{\beta_0}y_1^{\beta_1}\cdots y_n^{\beta_n}$$
这里 $a_{\alpha,\beta_0,\cdots,\beta_n}$ 为常数. 记 $\underline{\beta}=(\beta_0,\beta_1,\cdots,\beta_n)$ 以及
$$P_{\alpha,\underline{\beta}}=(\alpha-\beta_1-2\beta_2-\cdots-n\beta_n,\beta_0+\beta_1+\cdots+\beta_n)$$
那么由 $\{P_{\alpha,\underline{\beta}}|a_{\alpha,\underline{\beta}}\neq 0\}$ 生成的多边形被称为 $F=0$ 的 Newton 多边形, 记为 $\mathcal{N}(F)$. 令 $L(F,\mu)$ 为满足下面三个条件的一条直线：

N1. $L(F,\mu)$ 的斜率为 $-1/\mu$.

N2. $L(F,\mu)\cap\mathcal{N}(F)\neq\varnothing$.

N3. $\mathcal{N}(F)$ 位于 $L(F,\mu)$ 的右侧.

我们引进下面的多项式：

$$\Phi_{(F,\mu)}(C)=\sum_{P_{(\alpha,\underline{\beta})}\in L(F,\mu)}a_{\alpha\underline{\beta}}C^{\beta_0+\cdots+\beta_n}(\mu)_1^{\beta_1}\cdots(\mu)_n^{\beta_n}\tag{7.27}$$

这里 $(\mu)_k=\mu(\mu-1)\cdots(\mu-k+1)$. 那么我们需要用到下述引理 ([11])：

引理 7.3.3　利用前面的记号, 设 $y(x)=cx^{\mu}+\cdots$ 为微分方程 $F=0$ 的解. 则 $\Phi_{(F,\mu)}(c)=0$.

利用 Newton 多边形方法求 $F=0$ 在 $x=0$ 处的幂级数解的基本步骤如下：

1. $i:=n$ 以及 $\nu:=n-1$.
2. $\phi(x):=c_0+c_1x+\dfrac{c_2x^2}{2}+\cdots+\dfrac{c_{n-1}x^{n-1}}{(n-1)!}$ 以及 $\varphi(x):=\phi(x)$.
3. while $i\leqslant N$ do
   - i. $F:=F(y+\phi(x))$.
   - ii. 构造 Newton 多边形 $\mathcal{N}(F)$.
   - iii. 选择满足 N1, N2, N3 的直线$L(F,\,\mu)$, 并且其还满足$-1/\mu$为整数以及$-1/\mu>\nu$. 如果这样的 $L(F,\mu)$ 不存在, 则算法终止.
   - iv. 求非零的 c 使得它满足：$\Phi_{(F,\mu)}(c)=0$, 这里 $\Phi_{(F,\mu)}(c)$ 由 (7.27) 所定义. 如果这样的 c 不存在, 则回到 iii. 重新选择满足条件的 μ, 算法继续执行. 如果对于 iii. 中所有满足条件的 μ, 都不存在满足条件的 c, 则算法终止.
   - v. $\nu:=-1/\mu$, $\phi(x):=c*x^{\nu}$ 以及 $\varphi(x):=\phi(x)+\varphi(x)$.
   - vi. $i:=i+1$.
4. 返回 $\varphi(x)$.

例 7.3.4　考虑微分方程：$F=x^2+xy^2+y^3+y_1^2-2y_1+1=0$ 以及给定初值 $y(0)=0$. 我们有 $y_1(0)=1$, $S(0,0,1)=0$. 此时我们不能用第一种方法求幂级数解. 利用 Newton 多边形方法, 我们求得它的前面 5 项如下：

$$x+\frac{Ix^2}{2}+\frac{Ix^3}{3}-\Big(\frac{5}{16}+\frac{I}{8}\Big)x^4$$

这里 I 满足：$I^2+1=0$.

§7.4 微分方程的行波解

注释 7.4.1 求解数学物理方程, 尤其是在流体力学、空气动力力学、等离子体物理、生物物理和化学物理等学科领域引出的非线性发展方程, 是古老而在理论和实际上又很重要的课题. 但由于非线性方程的复杂性, 至今仍有大量的重要方程无法求出精确解, 即使已经求出精确解, 也各有技巧, 尚无一般的求解方法. 非线性发展方程的精确行波解可以很好地描述各种自然现象, 例如振动、传播波以及孤立子等, 因此它在非线性科学中起着非常重要的作用. 迄今为止, 已有多种构造精确行波解的有效方法, 如反散射方法 ([1])、Bäklund 变换法 ([73])、Darboux 变换法 ([47])、Hirota 双线性方法 ([49])、Painlevé 分析法 ([82]) 及 Lie 对称群方法 ([69]) 等. 近年来, 随着计算机和符号计算系统的发展, 构造非线性发展方程精确解的"直接法"越来越受到重视.

Tanh 函数方法 ([58]) 是构造非线性发展方程精确解的"直接法"之一, 是寻求一类特殊行波解 —— 扭状孤波解的一种非常有效的方法. 该方法是建立在大多数的孤波解都具有双曲函数形式的基础之上, 其本质在于对所求方程的解作了先验假设, 将其表示为双曲正切函数的多项式. 通过这样的假设, 将非线性发展方程的求解问题转化为非线性代数方程组的求解问题.

文献 [27] 与 [28] 对 Tanh 函数方法做了推广, 用 Riccati 方程的解代替 Tanh 函数, 并应用吴消元法求解所得代数方程组, 大大扩展了解的范围, 除孤波解外, 还可以得到周期解、有理解等等. 文献 [103] 进一步推广了文献 [27] 与 [28] 的算法, 除包含前者所得结果外, 还可以得到更多新解. 这里对文献 [103] 的算法稍做修改, 并给出其程序实现.

1. 算法叙述. 给定非线性发展方程 (组), 设其自变量为 $x=(t,\,x_1,\,x_2,\,\cdots,\,x_m)$, 未知函数为 $u=(u_1,\,u_2,\,\cdots,\,u_n)$, 其中 $m,\,n$ 为自然数, 求其具有如下形式的解:

$$u_k = a_{k\,0} + \sum_{i=1}^{N_k} \phi(\omega(x))^{i-1}\,[a_{k\,i}\,\phi(\omega(x)) + b_{k\,i}\,\Phi(\omega(x))] \tag{7.28}$$

其中

$$\omega(x) = x_1 + \sum_{j=2}^{m} c_{j-1}\,x_j + c_m\,t + c_{m+1},$$

$a_{k\,0},\ a_{k\,i},\ b_{k\,i}\ (i=1,2,\cdots,N_k,\ k=1,2,\cdots,n)$ 和 $c_j\ (j=1,2,\cdots,m+1)$ 是待定常数, ϕ 和 Φ 满足

$$\phi' = \delta + \phi^2, \quad \Phi^2 = \mu\,(\delta + \phi^2) \tag{7.29}$$

$\delta,\ \mu$ 是非零实数, “ $'$” 表示对 ω 求导, $1 \leqslant k \leqslant n$.

可以采取以下四个步骤来确定 u 的显式表达式：

1. 通过平衡已知方程的最高阶线性项和最高阶非线性项来确定平衡常数 N_k.
2. 由于 ϕ, Φ 满足 (7.29), 故有

$$\Phi' = \phi\,\Phi \tag{7.30}$$

将 (7.28) 代入已知方程 (组) 并用 (7.29), (7.30) 约化, 可得关于 $\phi,\ \Phi$ 的二元代数多项式方程 (组). 搜集各二元代数多项式的系数, 令这些系数为零可得到关于 $a_{k\,0},\ a_{k\,i},\ b_{k\,i}\ (i=1,2,\cdots,N_k,\ k=1,2,\cdots,n)$ 和 $c_j\ (j=1,2,\cdots,m+1)$ 的非线性代数方程组.

3. 用吴特征列方法求解步骤 2 中得到的代数方程组.
4. 由于 (7.29) 有如下精确解：

当 $\delta<0$ 时,

$$\phi = \frac{-\rho\sqrt{-\delta}\tanh\left(\sqrt{-\delta}\,\omega\right)-\epsilon\,\delta}{-\epsilon\sqrt{-\delta}\tanh\left(\sqrt{-\delta}\,\omega\right)+\rho},\quad \Phi = \frac{\operatorname{sech}\left(\sqrt{-\delta}\,\omega\right)\sqrt{\mu\,\delta\,\left(\epsilon^2\delta+\rho^2\right)}}{\epsilon\sqrt{-\delta}\tanh\left(\sqrt{-\delta}\,\omega\right)-\rho}$$

或

$$\phi = \frac{-\rho\sqrt{-\delta}\coth\left(\sqrt{-\delta}\,\omega\right)-\epsilon\,\delta}{-\epsilon\sqrt{-\delta}\coth\left(\sqrt{-\delta}\,\omega\right)+\rho},\quad \Phi = \frac{\operatorname{csch}\left(\sqrt{-\delta}\,\omega\right)\sqrt{-\mu\,\delta\,\left(\epsilon^2\delta+\rho^2\right)}}{\epsilon\sqrt{-\delta}\coth\left(\sqrt{-\delta}\,\omega\right)-\rho}$$

其中 ϵ,ρ 为任意常数, 且不同时为零.

当 $\delta>0$ 时,

$$\phi = \frac{\rho\sqrt{\delta}\tan\left(\sqrt{\delta}\,\omega\right)-\epsilon\,\delta}{\epsilon\sqrt{\delta}\tan\left(\sqrt{\delta}\,\omega\right)+\rho},\quad \Phi = \frac{\sec\left(\sqrt{\delta}\,\omega\right)\sqrt{\mu\,\delta\,\left(\epsilon^2\delta+\rho^2\right)}}{\epsilon\sqrt{\delta}\tan\left(\sqrt{\delta}\,\omega\right)+\rho}$$

其中 ϵ,ρ 为任意常数, 且不同时为零.

将 $\phi,\Phi,\ a_{k\,0},\ a_{k\,i},\ b_{k\,i}\ (i=1,2,\cdots,N_k,\ k=1,2,\cdots,n)$ 和 $c_j\ (j=1,2\cdots,m+1)$ 代入到 (7.28) 即可得到已知方程的显式精确解.

上述步骤 4 中只给出了感兴趣的孤波解和周期解. 事实上, 对 Riccati 方程

$$\phi' = \delta + \phi^2 \tag{7.31}$$

的任意一个解 ϕ, 相应地有 $\Phi = \pm\sqrt{\mu\,(\delta+\phi^2)}$, 进而可以利用上述算法, 通过解代数方程组求得非线性发展方程的精确解.

另外, Riccati 方程 (7.31) 有如下 Bäcklund 变换：

$$\tilde{\phi} = \frac{q\,\phi - \delta\,p}{p\,\phi + q} \tag{7.32}$$

即若 ϕ 是 (7.31) 的解, 则 $\tilde{\phi}$ 也是, 其中 p, q 为常数. 因此, 可利用 (7.31) 的已知解和 Bäcklund 变换 (7.32) 得到其新解.

Riccati 方程 (7.31) 还有一个解的非线性叠加公式：

$$\tilde{\phi} = \frac{\phi_1\,(\phi_3 - \phi_2) + k\,\phi_2\,(\phi_1 - \phi_3)}{\phi_3 - \phi_2 + k\,(\phi_1 - \phi_3)} \tag{7.33}$$

其中 k 为常数. 若已知 ϕ_1, ϕ_2, ϕ_3 为 (7.31) 的解, 则 $\tilde{\phi}$ 也是. 从 (7.31) 的任意三个已知解出发, 利用公式 (7.33), 可以得到其无穷序列解. 这样, 事实上可以利用上面的算法得到非线性发展方程的无穷序列解.

2. 程序与算例. 函数 twsolve 是上述算法的一个实现. 输入的非线性演化方程要求是未知函数及其导数的多项式形式, 系数可允许为任意常数, 即输入的方程中可包含自由参数. twsolve 将给出所输入的方程的系列精确解, 其中包含孤波解和周期波解. 如果所输入的方程包含参数, 在需要的情况下 twsolve 可以给出参数满足的条件以及在此种条件下方程的精确解.

函数 twsolve 的调用形式为 twsolve(ps, n). 其中参数 ps 是一个多项式列表. 多项式列表无序. n 是整数 0 或 1. 在所输入的方程包含参数的情况下, n 取 1 表示可以使参数满足某些限制；n 取 0 表示不允许参数有任何限制.

例 7.4.2 KdV-Burgers方程 $u_t + uu_x + pu_{xx} + qu_{xxx} = 0$ (允许参数满足某些限制条件) 的行波解.

在 MMP 中输入下面命令：

```
>read("de_twa.txt"): read("de_twb.txt"):
>depend([u], [x, t]):
>eqs:=[u[0, 1]+u*u[1, 0]+p*u[2, 0]+q*u[3, 0]]:
>twsolve(eqs, 1);
```

即可求得 KdV-Burgers 方程的系列行波解：

$$u_1 = -2 * \phi * p + a[0]$$

其中 $\phi = (-S * \sqrt{-Q} * \tanh(I * \sqrt{Q} * \xi) - Q * R)/(-R * \sqrt{-Q} * \tanh(I * \sqrt{Q} * \xi) + S)$, $\xi = x - a[0] * t + c[2], Q < 0$. 对于方程解中的参数, 我们有 $q = 0, c[2], a[0], p, K, Q, R, S$ 为自由参数.

$$u_2 = -2 * \phi * p + a[0]$$

其中 $\phi=(-S*\sqrt{-Q}*\coth(I*\sqrt{Q}*\xi)-Q*R)/(-R*\sqrt{-Q}*\coth(I*\sqrt{Q}*\xi)+S)$, $\xi=x-a[0]*t+c[2], Q<0$. 参数 $q=0$, $c[2], a[0], p, K, Q, R$ 及 S 为自由参数.

$$u_3=-2*\phi*p+a[0]$$

其中 $\phi=(S*\sqrt{Q}*\tan(\sqrt{Q}*\xi)-Q*R)/(R*\sqrt{Q}*\tan(\sqrt{Q}*\xi)+S)$, $xi=x-a[0]*t+c[2], Q>0$, 参数 $q=0$, $c[2], a[0], p, K, Q, R, S$ 为自由参数.

$$u_4=(-6*K*\phi^2*q+6*\sqrt{K}*\Phi*\phi*q+a[0]*K)/K$$

其中 ϕ, Φ 分别为 $(-S*\sqrt{-Q}*\tanh(I*\sqrt{Q}*\xi)-Q*R)/(-R*\sqrt{-Q}*\tanh(I*\sqrt{Q}*\xi)+S)$ 和 $(\mathrm{sech}(I*\sqrt{Q}*\xi)*(Q*K*(S^2+Q*R^2))^{(1/2)})/(R*\sqrt{-Q}*\tanh(I*\sqrt{Q}*\xi)-S)$, $\xi=x-5*Q*q*t-a[0]*t+c[2], Q<0$, 参数 $p=0$, $c[2], a[0], q, K, Q, R, S$ 为自由参数.

$$u_5=(-6*K*\phi^2*q+6*\sqrt{K}*\Phi*\phi*q+a[0]*K)/K$$

其中 ϕ, Φ 分别为 $(-S*\sqrt{-Q}*\coth(I*\sqrt{Q}*\xi)-Q*R)/(-R*\sqrt{-Q}*\coth(I*\sqrt{Q}*\xi)+S)$ 和 $(\mathrm{csch}(I*\sqrt{Q}*\xi)*(-Q*K*(S^2+Q*R^2))^{(1/2)})/(R*\sqrt{-Q}*\coth(I*\sqrt{Q}*\xi)-S)$, $\xi=x-5*Q*q*t-a[0]*t+c[2],\ Q<0$, 参数 $p=0$, $c[2], a[0], q, K, Q, R, S$ 为任意参数.

$$u_6=(-6*K*\phi^2*q+6*\sqrt{K}*\Phi*\phi*q+a[0]*K)/K$$

其中 ϕ, Φ 分别为 $(S*\sqrt{Q}*\tan(\sqrt{Q}*\xi)-Q*R)/(R*\sqrt{Q}*\tan(\sqrt{Q}*\xi)+S)$ 和 $(\sec(\sqrt{Q}*\xi)*(Q*K*(S^2+Q*R^2))^{(1/2)})/(R*\sqrt{Q}*\tan(\sqrt{Q}*\xi)+S)$, $xi=x-5*Q*q*t-a[0]*t+c[2],\ Q>0$, 参数 $p=0$, $c[2], a[0], q, K, Q, R, S$ 为自由参数.

twsolve 共给出 36 个解, 这里只列出其中 6 个.

例 7.4.3　Boussinesq方程 $u_t+u\,u_x+v_x=0,\ v_t+u_x\,v+u\,v_x+u_{xxx}=0$.

在 MMP 中输入下面命令：

```
>read("de_twa.txt"): read("de_twb.txt"):
>depend([u, v], [x, t]):
>eqs:=[u[0, 1]+u*u[1, 0]+v[1, 0], v[0, 1]+u[1, 0]*v+
       v[1, 0]*u+u[3, 0]]:
>twsolve(eqs, 0);
```

即可求得 Boussinesq 方程的如下行波解：

$$u_1=-2*\phi+a[0],\ \ v_1=-2*\phi^2-2*Q$$

其中 $\phi = (-S*\sqrt{-Q}*\tanh(I*\sqrt{Q}*\xi)-Q*R)/(-R*\sqrt{-Q}*\tanh(I*\sqrt{Q}*\xi)+S)$, $\xi = x-a[0]*t+f[2]$, $Q<0$, $f[2], a[0], K, Q, R, S$ 为自由参数.

$$u_2 = -2*\phi + a[0],\ \ v_2 = -2*\phi^2 - 2*Q$$

其中 $\phi = (-S*\sqrt{-Q}*\coth(I*\sqrt{Q}*\xi)-Q*R)/(-R*\sqrt{-Q}*\coth(I*\sqrt{Q}*\xi)+S)$, $\xi = x-a[0]*t+f[2]$, $Q<0$, $f[2], a[0], K, Q, R, S$ 为任意常数.

twsolve 共给出 36 个解, 这里只列出其中 2 个.

例 7.4.4 KP 方程 $(u_t + 6\,u\,u_x + u_{xxx})_x - u_{yy} = 0$.

在 MMP 中输入下面命令:

```
>read("de_twa.txt"): read("de_twb.txt"):
>depend([u], [x, y, t]):
>eqs:=[u[1, 0, 1]+6*u*u[2, 0, 0]+6*u[1, 0, 0]^2+
      u[4, 0, 0]-u[0, 2, 0]]:
>twsolve(eqs, 0);
```

即可求得 KP 方程的如下行波解: $u_1 = -2*\phi^2 + a[0]$, 其中 $\phi = (-S*\sqrt{-Q}*\tanh(I*\sqrt{Q}*\xi)-Q*R)/(-R*\sqrt{-Q}*\tanh(I*\sqrt{Q}*\xi)+S)$, $\xi = x-\sqrt{8*Q+6*a[0]+c[2]}*y+c[2]*t+c[3]$, $Q<0$, $c[2], c[3], a[0], K, Q, R, S$ 为自由参数.

$$u_2 = (-K*\phi^2 - \sqrt{K}*\Phi*\phi + a[0]*K)/K$$

其中 ϕ, Φ 分别为 $(-S*\sqrt{-Q}*\tanh(I*\sqrt{Q}*\xi)-Q*R)/(-R*\sqrt{-Q}*\tanh(I*\sqrt{Q}*\xi)+S)$ 和 $(\mathrm{sech}(I*\sqrt{Q}*\xi)*(Q*K*(S^2+Q*R^2))^{(1/2)})/(R*\sqrt{-Q}*\tanh(I*\sqrt{Q}*\xi)-S)$, $\xi = x+\sqrt{5*Q+6*a[0]+c[2]}*y+c[2]*t+c[3]$, $Q<0$, $c[2], c[3], a[0], K, Q, R, S$ 为任意常数.

这里列出 18 组解中的 2 组.

例 7.4.5 BLP方程 $2\,u\,u_{xy} + 2\,u_x\,u_y + 2\,v_{xxx} - u_{xxy} - u_{yt} = 0,\ 2\,u\,v_x + v_{xx} - v_t = 0$.

在 MMP 中输入下面命令:

```
>read("de_twa.txt"): read("de_twb.txt"):
>depend([u, v], [x, y, t]):
>eqs:=[2*u*u[1,1,0]+2*u[1,0,0]*u[0,1,0]+2*v[3,0,0]-u[2,1,0]-u[0,1,1],
      2*v[1, 0, 0]*u+v[2, 0, 0]-v[0, 0, 1]]:
>twsolve(eqs, 0);
```

即可求得 BLP 方程的如下行波解:

$$u_1 = -\phi + a[0],\ v_1 = c[1]*\phi + c[0]$$

其中 $\phi = (-S*\sqrt{-Q}*\tanh(I*\sqrt{Q}*\xi) - Q*R)/(-R*\sqrt{-Q}*\tanh(I*\sqrt{Q}*\xi)+S)$, $\xi = x - c[1]*y + 2*a[0]*t + f[3],\ Q < 0,\ f[3], a[0], c[1], c[0], K, Q, R, S$ 为自由参数.

$$u_2 = (-\sqrt{K}*\phi - \Phi + 2*\sqrt{K}*a[0])/(2*\sqrt{K}),\ v_2 = \sqrt{K}*d[1]*\phi + d[1]*\Phi + c[0]]$$

其中 ϕ, Φ 分别为 $(-S*\sqrt{-Q}*\tanh(I*\sqrt{Q}*\xi)-Q*R)/(-R*\sqrt{-Q}*\tanh(I*\sqrt{Q}*\xi)+S)$ 和 $(\mathrm{sech}(I*\sqrt{Q}*\xi)*(Q*K*(S^2+Q*R^2))^{(1/2)})/(R*\sqrt{-Q}*\tanh(I*\sqrt{Q}*\xi)-S)$, $\xi = x - 2*\sqrt{K}*d[1]*y + 2*a[0]*t + f[3],\ Q < 0,\ f[3], a[0], d[1], c[0], K, Q, R, S$ 为参数.

这里列出了 9 组解中的 2 组.

上面的四个算例分别为含参数 (1+1) 维方程、(1+1) 维方程组、(2+1) 维方程和 (2+1) 维方程组, 代表了一大类非线性发展方程.

文献说明

本章所采用的微分方程组的零点分解定理由吴文俊提出 ([100]). 关于微分预解式的工作请见 Ritt 的基本工作 ([72]) 与 Hubert 最近的工作 ([53]). 关于含参数的微分方程的零点定理来自 [40] 和 [81]. 微分有理参数方程的隐式化结果来自 [32]. 常微分方程代数函数解与有理函数解的结果来自 [30] 和 [2]. 求微分方程的 Puiseux 幂级数解的算法请参考 [11]. 求偏微分方程组的升列的幂级数解的算法请参考 [100]. 文献 [27] 和 [28] 对 Tanh 函数方法做了推广, 求偏微分方程行波解的算法来自 [27],[28] 和 [103].

第八章　代数方程组的实数解与不等式机器证明

代数方程实数解的求解比复数域上的方程求解要困难的多. 主要原因是复数解的一些基本性质可以由其方程的形式确定, 而在实数情形这一确定过程则要困难的多. 另一方面, 很多具体的应用例如, 机器人、计算机视觉, 都要求我们给出或判定方程组的实解. 本章将介绍怎样用 MMP 进行方程实解的求解与不等式机器证明.

§8.1　代数方程的实根隔离

1. 单个方程的实根隔离.　处理实数与代数数的一个基本问题是像 $\sqrt{2}$ 这样的数怎样在计算机中唯一表示. 例如 $\sqrt{2}$ 可以用其满足的极小多项式与一个充分小的区间唯一表示：

$$x^2-2=0,\quad (0,2)$$

在此基础上, 我们可以设计计算机软件进行代数数的加、减、乘、除等算术运算. 这里的一个核心算法是方程的实根隔离算法.

假定我们所考虑的方程是不可约的. 否则的话, 可以用 MMP 将其分解为不可约方程的乘积, 然后再考虑这些不可约方程. 设 $P(x)$ 是一个不可约多项式, 具有实根：$s_1<s_2<\cdots<s_m$. 我们可以求得有理数 $r_{i,l}, r_{i,r}$, 使得

$$r_{1,l}<s_1<r_{1,r}<r_{2,l}<s_2<r_{2,r},\cdots,r_{m,l}<s_m<r_{m,r} \tag{8.1}$$

则 $P(x)=0$ 的实根 s_i 可以由

$$P(x)=0 \text{ 与有理区间} (r_{i,l}, r_{i,r})$$

唯一确定. 我们称满足 (8.1) 的有理数对 $(r_{i,l}, r_{i,r})$ 为方程 $P(x)=0$ 的一组 实根隔离. 如果 $r_{i,r}-r_{i,l}$ 的值非常小, 我们可以用 $\dfrac{r_{i,r}+r_{i,l}}{2}$ 作为 s_i 的近似解.

在实根隔离算法中, 下列古典结果起着重要作用.

引理 8.1.1 (Descartes 符号法则[66])　*设 $f=\sum_{i=k}^n a_ix^i, (a_na_k\neq 0)$ 是一个实系数多项式. 则 $f=0$ 的正实根个数 N 不超过系数序列 $a_n,\cdots,a_k$ 的符号变化个数 v, 而且 $v-N$ 是一个偶数.*

MMP 的函数 realroot(poly, wd) 实现了上面所提的单变元多项式的实根隔离算法. 其中 poly 是一个单变元多项式, wd 是一个正数. realroot 函数基于 Descartes

符号法则隔离单变元多项式 poly 的所有实根. 函数返回值是隔离区间的列表, 输出区间的宽度小于等于 wd. 输出区间从小到大排序. 重根只输出一次.

```
>realroot((x-1)^2*x*(x+3), 1);
```

$[[-3,-3],[0,0],[1,1]]$

由这一例子可以看到, 如果方程 poly=0 有有理解, 则 realroot 输出以这个有理解为左右边界的区间.

我们可以通过改变 wd 的值来得到方程实根的近似值.

```
>realroot(3*x^10-9*x^3+x-1, 1/10);
```

$\left[\left[-\frac{9}{16},-\frac{1}{2}\right],\left[\frac{9}{8},\frac{19}{16}\right]\right]$

```
>realroot(3*x^10-9*x^3+x-1, 1/100);
```

$\left[\left[-\frac{9}{16},-\frac{71}{128}\right],\left[\frac{149}{128},\frac{75}{64}\right]\right]$

上例说明, 输入多项式具有两个不同的实根. 而且, 这两个实根可以近似取为 $-\frac{143}{256},\frac{299}{256}$.

2. 方程组的实根隔离. 对于一个给定的方程组, 如果其存在实根, 根据第四章, 利用吴特征列方法, 我们可以将原方程组的零点在一定的变量序下分解为一组升列的正则零点的并. 特别的, 我们也可以要求该升列是饱和的, 即升列中多项式的初式和隔离子对于其部分升列是可逆的. 我们进一步假定, 对应的饱和升列是零维的. 从而我们可以将多变量多项式组的求解转化为单变量多项式的求解. 利用单变量的实根分离算法 realroot, 可以构造多变量的实根分离算法 ([65]).

下面以两个变量的情形概述一下该算法的思想. 已知饱和升列 $A_1 = f(x)$, $A_2 = g(x,y)$, 首先由 realroot 算法, 求出 $f(x)$ 的实根分离区间解：$[a_1,b_1],\cdots,[a_n, b_n]$. 任取 $[a_i,b_i]$, 记为 $[a,b]$, 不妨设 $a>0$, 我们知道有且仅有一个 $x^*\in[a,b]$, 满足 $f(x^*)=0$. 在 $g(x,y)$ 中, 按照其常系数的符号, 将其分为两个部分 g_+ 和 g_-, $g(x,y)=g_+(x,y)+g_-(x,y)$. 令 $\overline{g}(y)=g_+(b,y)+g_-(a,y)$, $\underline{g}(y)=g_+(a,y)+g_-(b,y)$, 对于 $y>0$, 显然有 $\underline{g}(y)\leqslant g(x^*,y)\leqslant\overline{g}(y)$. 见图 8.1.

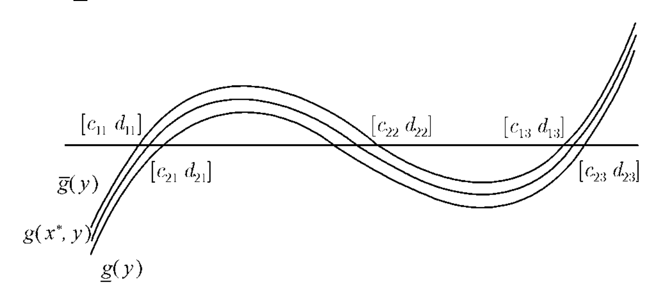


图 8.1　极大、极小多项式以及 $g(x^*,y)$

由 realroot, 我们可以分别求出 $\underline{g}(y)$ 以及 $\overline{g}(y)$ 的正实根, 如图 8.1 中的 $[c_{1i}, d_{1i}]$ 以及 $[c_{2i}, d_{2i}]$, 对其进行匹配, 可以得到 $g(x^*, y)$ 的正实根 $(x^*, y^*) \in [[a, b], [c_i, d_i]]$, 其中 $c_i = \min(c_{1i}, c_{2i})$, $d_i = \min(d_{1i}, d_{2i})$. 对于负的实根, 可以通过做变量替换 $y = -y$ 转化为正实根的情形得到. 求出实根的分离区域后, 通过判断升列中多项式的隔离子在各个实根分离区域中的符号, 确定每个区间中解的唯一性. 因为升列是饱和的, 在精度足够的情况下, 总可以保证在每个区域中有且仅有原多项式组的一个实根 (不计重数).

在 MMP 中, 函数 Mrealroot(polyset, vars, precision) 实现了多元多项式的实根分离算法. 其中参数 polyset 是多项式列表, vars 是变量序, precison 是预先给定的区间宽度. Mrealroot 的返回值是 polyset 的所有实根, 其实根的形式是一组区间, 构成空间中的一个矩形区域, 每个区间对应一个变量. 在该矩形区域中有且只有原多项式组的一个实根. 重根只输出一次. 通过提高精度, 即令 precision 趋于 0, 则每个区间的左右端点趋于同一点, 该点就是实根.

在 MMP 的用户界面中输入：

```
>Mrealroot([x-2, y^2+x-6, z^2+x+y-10], [x, y, z], 1/1000);
```

输出结果为

$$\left[[2,2],[-2,-2],\left[\frac{1619}{512},\frac{3239}{1024}\right]\right];$$

$$\left[[2,2],[-2,-2],\left[\frac{-3239}{1024},\frac{-1619}{512}\right]\right];$$

$$\left[[2,2],[2,2],\left[\frac{-2509}{1024},\frac{-627}{256}\right]\right];$$

$$\left[[2,2],[2,2],\left[\frac{627}{256},\frac{2509}{1024}\right]\right].$$

3. 微分多项式系统的稳定性. 根据微分方程定性理论, 利用单调系统理论可以将单调微分多项式系统的永久性及稳定性问题化为多项式系统的实根分布以及根的正 (负) 性判定问题. 在 [65] 中, 利用 Mrealroot 算法解决了几类单调 Lotka-Volterra 扩散系统的永久性及稳定性问题. 这里, 作为例子, 我们利用 MMP 中的 Mrealroot 命令计算出几类系统的实根.

例 8.1.2　具有多个正平衡点的永久生存系统

$$\begin{aligned}
\dot{x}_1 &= x_1(13 - 130x_1 + 31x_2) + 12(y_1 - x_1) \\
\dot{x}_2 &= x_2(13 + 531x_1 - 130x_2) + 231(y_2 - x_2) \\
\dot{y}_1 &= y_1(13 - 130y_1 + 531y_2) + 231(x_1 - y_1) \\
\dot{y}_2 &= y_2(13 + 31y_1 - 130y_2) + 12(x_2 - y_2)
\end{aligned} \tag{8.2}$$

这是一个具有永久生存性的系统, 通过求解 (8.2) 右端的多项式组的实根, 我们知道这个系统有三个正平衡点, 由于正平衡点的不唯一, 所以系统不具有全局稳定性. 在MMP中, 利用Mrealroot 命令, 我们可以给出系统 (8.2) 的六组实根 (实根分离区间的精度为 10^{-10}, 变元序为 $[x_1, x_2, y_1, y_2]$):

```
>ps:=[x1*(13-130*x1+31*x2)+12*(y1-x1),x2*(13+531*x1-130*x2)+
      231*(y2-x2),y1*(13-130*y1+531*y2)+231*(x1-y1),y2*(13+31*y1-
      130*y2)+12*(x2-y2)]:
>Mrealroot(ps,[x1,x2, y1, y2],1/10000000000);
```

输出结果为

$$\Big[[2,2],[7,7],[7,7],[2,2]\Big];$$
$$\Big[[1,1],[3,3],[3,3],[1,1]\Big];$$
$$\Big[[0,0],[0,0],[0,0],[0,0]\Big];$$
$$\Big[\Big[\frac{138}{439},\frac{138}{439}\Big],\Big[\frac{253}{439},\frac{253}{439}\Big],\Big[\frac{253}{439},\frac{253}{439}\Big],\Big[\frac{138}{439},\frac{138}{439}\Big]\Big];$$
$$\Big[[0,0],\Big[\frac{1}{10},\frac{1}{10}\Big],[0,0],\Big[\frac{1}{10},\frac{1}{10}\Big]\Big];$$
$$\Big[\Big[\frac{1}{10},\frac{1}{10}\Big],[0,0],\Big[\frac{1}{10},\frac{1}{10}\Big],[0,0]\Big].$$

Goh 于 1980 曾给出两个竞争扩散系统, 并且指出要证明系统是全局稳定的, 只要证明相应系统只有一个正平衡点. 在 MMP 中, 利用 Mrealroot 命令, 我们可以计算出相应系统的所有平衡点, 发现其中确实只有一个正平衡点, 从而验证了原有的结果. 这就是下面的两个例子.

例 8.1.3　考虑微分多项式系统

$$\begin{aligned}\dot{x}_1&=x_1(4-x_1-y_1)+1/10(x_2-x_1)\\ \dot{x}_2&=x_2(1-x_2-y_2)+1/10(x_1-x_2)\\ \dot{y}_1&=y_1(3-2x_1-y_1)+1/10(y_2-y_1)\\ \dot{y}_2&=y_2(3-2x_2-y_2)+1/10(y_1-y_2)\end{aligned}\tag{8.3}$$

在MMP 中, 利用Mrealroot 命令得到系统右端多项式组的 12组实解, 其中恰好存在唯一一组正解, 保证了系统 (8.3) 的全局稳定性. Mrealroot 输出的十二组实根 (实根分离区间的精度为 10^{-15}, 变元序为 $[x_1, x_2, y_1, y_2]$) 为

```
>ps:=[x1*(4-x1-y1)+(1/10)*(x2-x1),x2*(1-x2-y2)+(1/10)*(x1-x2),
      y1*(3-2*x1-y1)+(1/10)*(y2-y1),y2*(3-2*x2-y2)+(1/10)*(y1-y2)]:
>Mrealroot(ps,[x1,x2, y1, y2],1/1000000000000000);
```

输出结果为

$[[0,0],[0,0],[0,0],[0,0]];$

$[[0,0],[0,0],[3,3],[3,3]];$

$$\Big[[0,0],[0,0],\Big[\frac{407669068857749}{140737488355328},\frac{203834534428875}{70368744177664}\Big],\Big[\frac{-13604101462861}{140737488355328},\frac{-13604101462773}{140737488355328}\Big]\Big];$$

$$\Big[[0,0],[0,0],\Big[\frac{-13604101462831}{140737488355328},\frac{-6802050731415}{70368744177664}\Big],\Big[\frac{407669068857729}{140737488355328},\frac{407669068857761}{140737488355328}\Big]\Big];$$

$$\Big[\Big[\frac{547721868534657}{140737488355328},\frac{273860934267329}{70368744177664}\Big],\Big[\frac{-11231107186391}{35184372088832},\frac{-22462212389829}{70368744177664}\Big],[0,0],[0,0]\Big];$$

$$\Big[\Big[\frac{553250243089995}{140737488355328},\frac{138312560772499}{35184372088832}\Big],\Big[\frac{42986727428605}{35184372088832},\frac{171946944584427}{140737488355328}\Big],[0,0],[0,0]\Big];$$

$$\Big[\Big[\frac{-3219702453095}{140737488355328},\frac{-1609851226547}{70368744177664}\Big],\Big[\frac{31576218231985}{35184372088832},\frac{63152543610771}{70368744177664}\Big],[0,0],[0,0]\Big];$$

$$\Big[\Big[\frac{542510614545557}{140737488355328},\frac{271255307272779}{70368744177664}\Big],\Big[\frac{30371384360973}{140737488355328},\frac{30371384651321}{140737488355328}\Big],\Big[\frac{7153476332849}{140737488355328},\frac{894185721547}{17592186044416}\Big],\Big[\frac{173842754025609}{70368744177664},\frac{347685509144881}{140737488355328}\Big]\Big];$$

$$\Big[\Big[\frac{557052462336791}{140737488355328},\frac{69631557792099}{17592186044416}\Big],\Big[\frac{250453804619847}{140737488355328},\frac{250453873462795}{140737488355328}\Big],\Big[\frac{-924380602149}{70368744177664},\frac{-1848485130487}{140737488355328}\Big],\Big[\frac{-46243830392309}{70368744177664},\frac{-180639954825}{274877906944}\Big]\Big];$$

$$\Big[\Big[\frac{-1797174237367}{35184372088832},\frac{-7188696949467}{140737488355328}\Big],\Big[\frac{281358976417039}{140737488355328},\frac{8792524747181}{4398046511104}\Big],\Big[\frac{5227808220309}{140737488355328},\frac{5231311035365}{140737488355328}\Big],\Big[\frac{-155055835178941}{140737488355328},\frac{-77527820558187}{70368744177664}\Big]\Big];$$

$$\Big[\Big[\frac{-132256146767387}{140737488355328},\frac{-66128073383693}{70368744177664}\Big],\Big[\frac{-3009847135523}{70368744177664},\frac{-3009847134081}{70368744177664}\Big],\Big[\frac{340886461839919}{70368744177664},\frac{85221615530989}{17592186044416}\Big],\Big[\frac{110472948345065}{35184372088832},\frac{441891793784553}{140737488355328}\Big]\Big];$$

$$\left[\left[\frac{-17471810856111}{17592186044416}, \frac{-139774486848887}{140737488355328}\right], \left[\frac{7221253451127}{70368744177664}, \frac{3610626727031}{35184372088832}\right],\right.$$
$$\left.\left[\frac{171799122522317}{35184372088832}, \frac{343598245274211}{70368744177664}\right], \left[\frac{-23984416128519}{140737488355328}, \frac{-23984415993973}{140737488355328}\right]\right].$$

例 8.1.4 考虑微分多项式系统

$$\begin{aligned}\dot{x}_1 &= x_1(2 - x_1 - y_1) + 1/2(x_2 - x_1)\\ \dot{x}_2 &= x_2(2 - x_2 - y_2) + 1/2(x_1 - x_2)\\ \dot{y}_1 &= y_1(5 - x_1 - 2y_1) + 1/2(y_1 - y_2)\\ \dot{y}_2 &= y_2(3/2 - x_2 - 2y_2) + 1/2(y_2 - y_1)\end{aligned} \tag{8.4}$$

在 MMP 中, 利用Mrealroot 命令得到系统右端多项式组的 14 组实解, 其中恰好存在唯一一组正解, 保证了系统 (8.4) 的全局稳定性. Mrealroot 输出的十四组实根 (实根分离区间的精度为 10^{-15}, 变元序为 $[x_1, x_2, y_1, y_2]$) 为

```
>ps:=[x1*(2-x1-y1)+(1/2)*(x2-x1),x2*(2-x2-y2)+(1/2)*(x1-x2),
y1*(5-x1-2*y1)+(1/2)*(y1-y2),y2*((3/2)-x2-2*y2)+(1/2)*(y2-y1)]:
>Mrealroot(ps,[x1,x2, y1, y2],1/1000000000000000);
```

输出结果为

$[[2, 2], [2, 2], [0, 0], [0, 0]];$

$[[0, 0], [0, 0], [0, 0], [0, 0]];$

$$\left[[0, 0], [0, 0], \left[\frac{18806586567}{8589934592}, \frac{37613173135}{17179869184}\right], \left[\frac{-9121220423}{17179869184}, \frac{-9121220395}{17179869184}\right]\right];$$

$$\left[[0, 0], [0, 0], \left[\frac{40578689031}{17179869184}, \frac{5072336129}{2147483648}\right], \left[\frac{2272215285}{2147483648}, \frac{18177722309}{17179869184}\right]\right];$$

$$\left[[0, 0], [0, 0], \left[\frac{-441225419}{8589934592}, \frac{-882450837}{17179869184}\right], \left[\frac{4061683645}{8589934592}, \frac{8123367301}{17179869184}\right]\right];$$

$$\left[\left[\frac{23468137739}{17179869184}, \frac{5867034435}{4294967296}\right], \left[\frac{-6288269923}{17179869184}, \frac{-6288267189}{17179869184}\right], [0, 0], [0, 0]\right];$$

$$\left[\left[\frac{-1572067139}{4294967296}, \frac{-6288268555}{17179869184}\right], \left[\frac{11734067829}{8589934592}, \frac{23468139523}{17179869184}\right], [0, 0], [0, 0]\right];$$

$$\left[\left[\frac{8106745829}{17179869184}, \frac{4053372915}{8589934592}\right], \left[\frac{17378473291}{17179869184}, \frac{17378474161}{17179869184}\right],\right.$$
$$\left.\left[\frac{18038672247}{8589934592}, \frac{2254834215}{1073741824}\right], \left[\frac{6199190097}{8589934592}, \frac{12398380369}{17179869184}\right]\right];$$

$$\left[\left[\frac{22008742687}{17179869184}, \frac{687773209}{536870912}\right], \left[\frac{25574701505}{8589934592}, \frac{51150815681}{17179869184}\right],\right.$$
$$\left.\left[\frac{23723842345}{17179869184}, \frac{5931423535}{4294967296}\right], \left[\frac{-21684686823}{17179869184}, \frac{-10841906509}{8589934592}\right]\right];$$

$$[[\frac{6099143733}{4294967296},\frac{24396574933}{17179869184}],[\frac{-9318385897}{17179869184},\frac{-4659190145}{8589934592}],$$
$$[\frac{-1907752257}{17179869184},\frac{-953859733}{8589934592}],[\frac{3149642793}{4294967296},\frac{12598972903}{17179869184}]];$$
$$[[\frac{33958849603}{17179869184},\frac{8489712401}{4294967296}],[\frac{12121980961}{4294967296},\frac{757628031}{268435456}],$$
$$[\frac{4075894471}{17179869184},\frac{4076247345}{17179869184}],[\frac{-8351961753}{8589934592},\frac{-16700581181}{17179869184}]];$$
$$[[\frac{-8002549887}{17179869184},\frac{-4001274943}{8589934592}],[\frac{15386091163}{8589934592},\frac{30772190629}{17179869184}],$$
$$[\frac{741494609}{17179869184},\frac{185375419}{4294967296}],[\frac{-7236263387}{17179869184},\frac{-3618131235}{8589934592}]];$$
$$[[\frac{-7118955485}{4294967296},\frac{-28475821939}{17179869184}],[\frac{8001664513}{17179869184},\frac{8001664583}{17179869184}],$$
$$[\frac{25915933101}{8589934592},\frac{1619745843}{536870912}],[\frac{-12801181381}{17179869184},\frac{-12801181115}{17179869184}]];$$
$$[[\frac{-19735498591}{8589934592},\frac{-39470997181}{17179869184}],[\frac{39146449581}{17179869184},\frac{19573226367}{8589934592}],$$
$$[\frac{56721491315}{17179869184},\frac{56721501041}{17179869184}],[\frac{-11018898877}{8589934592},\frac{-22037796955}{17179869184}]].$$

§8.2 代数系统全局优化的吴有限核定理

1. 代数系统的优化问题. 很多关于实数的问题可以归结为如下的极值问题：

> 设 $\mathbb{R}$ 是实数域, $\mathbb{D}$ 是 $\mathbb{R}^n$ 中的一个区域, f, g 是 $\mathbb{R}[x_1,\cdots,x_n]$ 中的多项式, 而 $\mathbb{H}$ 是 $\mathbb{R}[x_1,\cdots,x_n]$ 中的一组多项式, 要确定 f 在约束条件 $\mathbb{H}=0$ 与 $g\neq 0$ 下的极值.

首先, 我们定义一些基本概念. 设 $g\in\mathbb{R}[\mathbb{X}]$ 是一个实多项式, $\mathbb{H}\subset\mathbb{R}[\mathbb{X}]$ 是一个实多项式集合, 在本节, 我们用 $\mathrm{Zero}(g)$ 表示 g 的所有*实数零点*. $\mathrm{Zero}(\mathbb{H})$ 表示实多项式组 $\mathbb{H}$ 的所有的实公共零点. $\mathrm{Zero}(\mathbb{H}/g)$ 定义为 $\mathrm{Zero}(\mathbb{H})\backslash\mathrm{Zero}(g)$.

$X^0\in\mathbb{R}^n$ 是 f 在约束条件 $\mathbb{H}=0$, $g\neq 0$ 下, 在区域 $\mathbb{D}$ 中的一个*极值零点*, 如果满足：存在一个包含 X^0 的邻域 $V\subset\mathbb{R}^n$, 对任意的 $X'\in V\cap\mathbb{D}\cap\mathrm{Zero}(\mathbb{H}/g)$, 有 $f(X')\leqslant f(X^0)$, 或者对任意的 $X'\in V\cap\mathbb{D}\cap\mathrm{Zero}(\mathbb{H}/g)$, 有 $f(X')\geqslant f(X^0)$. 所有这样的极值零点构成的集合记为 $\mathrm{EZero}\{f,g,\mathbb{D},\mathbb{H}\}$.

如果 X^0 是在约束条件 $\mathbb{H}=0$, $g\neq 0$ 下, f 在区域 $\mathbb{D}$ 中的一个极值零点, 那么我们称实数 $r=f(X^0)$ 是在约束条件 $\mathbb{H}=0$, $g\neq 0$ 下, f 在区域 $\mathbb{D}$ 中的一个*极值*.

所有这样的极值构成的集合记为 $\mathrm{EVal}\{f,g,\mathbb{D},\mathbb{H}\}$.

对于上述的极值问题, 设 $\mathbb{H}=\{h_1,\cdots,h_m\}$, 我们引进 Lagrange *多项式*：

$$L=f+\sum_{j=1}^{m}\lambda_j h_j$$

多项式组 $\left\{\frac{\partial L}{\partial x_i},h_j|i=1,\cdots,n,j=1,\cdots,m\right\}\subset\mathbb{R}[x_1,\cdots,x_n,\lambda_1,\cdots,\lambda_m]$ 则称为 Lagrange *多项式组*, 记为 $\mathbb{L}_{(f,\mathbb{H})}$. 对于 $\mathbb{R}[x_1,\cdots,x_n,\lambda_1,\cdots,\lambda_m]$ 中的多项式组 $\mathbb{L}_{(f,\mathbb{H})}$, 在与上面的记号 Zero 的意义不混淆的情况下, 我们用 $\mathrm{Zero}(\mathbb{L}_{(f,\mathbb{H})})$ 来表示集合

$$\{(x_1,\cdots,x_n,\lambda_1,\cdots,\lambda_m)\in\mathbb{R}^{n+m}|l(x_1,\cdots,x_n,\lambda_1,\cdots,\lambda_m)=0,\forall l\in\mathbb{L}_{(f,\mathbb{H})}\}$$

与上面 Lagrange 多项式相关的 Lagrange *投影映射* 定义为 $\mathrm{LProj}:\mathbb{R}^{n+m}\to\mathbb{R}^n$. 对于 $(a_1,\cdots,a_n,b_1,\cdots,b_m)\in\mathbb{R}^{n+m}$,

$$\mathrm{LProj}(a_1,\cdots,a_n,b_1,\cdots,b_m)=(a_1,\cdots,a_n)$$

设 $\mathbb{H}=\{h_1,\cdots,h_m\}$, 对 $1\leqslant i_1\leqslant\cdots\leqslant i_m\leqslant n$, 由 $(i_1,\cdots,i_m)$ 决定的 Jacobi*行列式*定义为

$$\frac{\partial(h_1,\cdots,h_m)}{\partial(x_{i_1},\cdots,x_{i_m})}=\left|\frac{\partial h_j}{\partial x_{i_k}}\right|$$

由 $\mathbb{H}$ 的所有 Jacobi 行列式构成的集合与 $\mathbb{H}$ 的并称为 Jacobi*多项式组*, 记为 $\mathbb{J}_{\mathbb{H}}$. 关于极值的一个基本结果是

定理 8.2.1 *对于 $\mathbb{D}$ 是开集的情形, $\mathbb{H}$ 是 $\mathbb{R}[x_1,\cdots,x_n]$ 中的一组多项式, $\mathbb{L}_{(f,\mathbb{H})}$ 是由 $\mathbb{H}$ 所定义的Lagrange多项式集, $\mathbb{J}_{\mathbb{H}}$ 是由 $\mathbb{H}$ 所定义的Jacobi多项式集, 我们有*

$$\mathrm{EZero}\{f,g,\mathbb{D},\mathbb{H}\}\subset\mathrm{LProj}(\mathrm{Zero}(\mathbb{L}_{(f,\mathbb{H})}/g))\cup\mathrm{Zero}(\mathbb{J}_{\mathbb{H}}/g)$$

设 $\mathbb{H}$ 是含变量 $x_1,\cdots,x_n$ 的多项式集合. 引入一新变量 x_0, 安排其序 $x_0\prec x_1$. 引进一个新的多项式 $h_0=f-x_0\in\mathbb{R}[x_0,x_1,\cdots,x_n]$, 令 $\mathbb{H}'=\mathbb{H}\cup\{h_0\}$. 同时, g 自然地可以看作是 $\mathbb{R}[x_0,x_1,\cdots,x_n]$ 中的一个多项式. 设 $\mathbb{H}'$ 是 $\mathbb{R}[x_0,x_1,\cdots,x_n]$ 中的多项式, 为简单起见, 我们仍然用 $\mathrm{Zero}(\mathbb{H}')$ 来表示集合 $\{(x_0,x_1,\cdots,x_n)\in\mathbb{R}^{n+1}|h(x_0,x_1,\cdots,x_n)=0,\forall h\in\mathbb{H}'\}$. 用 $\mathbb{D}'$ 来表示集合 $\{(x_0,x_1,\cdots,x_n)\in\mathbb{R}^{n+1}|x_0\in\mathbb{R},(x_1,\cdots,x_n)\in\mathbb{D}\}$.

定义 Proj_0 是 $\mathbb{R}^{n+1}$ 到 $\mathbb{R}$ 的自然投影映射, 对 $(a_0,a_1,\cdots,a_n)\in\mathbb{R}^{n+1}$, 有 $\mathrm{Proj}_0(a_0,a_1,\cdots,a_n)=a_0$. 我们有下面的结果：

引理 8.2.2 设 f, g 是 $\mathbb{R}[x_1, \cdots, x_n]$ 中的多项式, $\mathbb{H}$ 是 $\mathbb{R}[x_1, \cdots, x_n]$ 中的多项式组, $\mathbb{D}'$, $\mathbb{H}'$ 如上所述, 则

$$\{f(x_1, \cdots, x_n)|(x_1, \cdots, x_n) \in \mathrm{Zero}(\mathbb{H})\} = \mathrm{Proj}_0(\mathrm{Zero}(\mathbb{H}'))$$

及

$$\mathrm{EVal}\{f, g, \mathbb{D}, \mathbb{H}\} = \mathrm{EVal}\{x_0, g, \mathbb{D}', \mathbb{H}'\}$$

定理 8.2.3 (开区域上的有限核定理) 设 $\mathbb{D}$ 是一个开区域, 令 $\mathbb{H}$ 是多项式集, f 是 $\mathbb{R}[x_1, \cdots, x_n]$ 中任一多项式. 那么有一有限实数集 $\mathbb{K}$ 使得

$$\mathrm{EVal}\{f, \mathbb{D}, \mathbb{H}\} \subset \mathbb{K} \subset \mathrm{Proj}_0(\mathrm{Zero}(\mathbb{H}'))$$

上面定理中的集合 $\mathbb{K}$ 称为极值问题 f 在区域 $\mathbb{D}$ 中及约束条件 $\mathbb{H} = 0$ 下的有限核, 记为 $\mathbb{K}(f, \mathbb{D}, \mathbb{H})$. 同时, f 的极值可以在有限集合 $\mathbb{K}$ 中取到. 由上述定理, 再利用第四章中介绍的计算投影的算法, 就可以计算有限核, 从而求得代数系统的极值.

上面的定理中, 如果有条件 $g \neq 0$, 定理仍然正确. 还需要注意的是, 虽然 $\mathrm{EZero}\{f, g, \mathbb{D}, \mathbb{H}\}$ 可能是无限集, 但 $\mathrm{EVal}\{f, g, \mathbb{D}, \mathbb{H}\}$ 是一有限集.

Lagrange 乘子法可对任何可微函数 f, h_j 适用. 然而上述定理所隐含的有限性质却不再成立. 这可由例子 $f = x\ \sin\left(\frac{1}{x}\right), \mathbb{D} = \{x > 0\}$ 看出. 因此, 极值点集合的有限性是 f, h_j 为多项式时的特征.

设 $\mathbb{R}$ 是实数域, $\mathbb{D}$ 是 $\mathbb{R}^n$ 中的一个区域, f, g 是 $\mathbb{R}[x_1, \cdots, x_n]$ 中的多项式, 而 $\mathbb{H}$ 是 $\mathbb{R}[x_1, \cdots, x_n]$ 中的一组多项式, 要确定 f 在约束条件 $\mathbb{H} = 0$, $g \neq 0$ 下的极值.

对于这一极值问题, 计算过程描述如下. 首先引进新变元 x_0, 令 $f' = f - x_0$, 设 $\mathbb{H}' = \mathbb{H} \cup \{f'\}$, 对于给定的变元序 $x_0 < x_1 < \cdots < x_n$, 我们计算 $\mathbb{H}'$ 的如下形式的零点分解, 在有限步内得到有限个升列 $\mathcal{A}_1, \cdots, \mathcal{A}_s$, 使得

$$\mathrm{Zero}(\mathbb{H}') = \cup_{i=1}^{s} \mathrm{Zero}(\mathcal{A}_i / I_{\mathcal{A}_i} S_{\mathcal{A}_i})$$

令 $K = \cup_{i=1}^{s} K_i$, 其中 $K_i = \mathrm{Proj}_0 \mathrm{Zero}(\mathcal{A}_i / I_{\mathcal{A}_i} S_{\mathcal{A}_i})$. K 就是我们要求的有限核, 需要求的极值一定在有限集合 K 中可以取到.

MMP 系统的函数EVal实现了上述有限核定理. 下面我们来介绍 EVal 的调用方法.

```
EVal(H, Y)
EVal(H, Y, D)
EVal(H, Y, D, type)
```

参数: H - 变元为Y的多项式的列表.

Y – 变元表(不包含参数).

D – 变元为Y的多项式的列表.

type – 只能取 "ritt", "wu", "weak". 缺省值为"ritt".

输出: 一组升列构成的列表. 这些升列的零点定义了有限核.

如果输出是 $[\mathcal{A}_1, \cdots, \mathcal{A}_r]$, 那么有

$$\mathrm{EZero}\{f, \mathbb{D}, \mathbb{H}\} = \cup_{i=1}^{r} \mathrm{Zero}(\mathcal{A}_i / I_{\mathcal{A}_i} S_{\mathcal{A}_i} \mathbb{D})$$

而且 $\mathcal{A}_i$ 是参数 type 指定形式的升列. 如果 $\mathcal{A}_i$ 中有仅含有 x_0 的多项式, 记为 r_i. 否则令 $r_i = 1$. 则有限核是 $r_i = 0$ 解的子集合.

例 8.2.4 计算 $f(x, y) = x^2 + y^2$, 在约束条件 $h_1 = (x - a)^3 - y^3 = 0$ 下的极值.

根据上面介绍的方法, 我们引进一个新变量 x_0 并取 $h_0 = x^2 + y^2 - x_0$. 那么 $\mathbb{H}' = \{h_0, h_1\}$. 在变元序 $x_0 < x < y$ 下, 容易发现, 我们得到如下形式的零点分解:

$$\mathrm{Zero}(\mathbb{H}') = \cup_{i=1}^{3} \mathrm{Zero}(\mathcal{A}_i / I_{\mathcal{A}_i} S_{\mathcal{A}_i})$$

其中

$\mathcal{A}_1 = \{x^3 - 4x^2 + 3x + x_0 - 1, x^2 + y^2 - x_0\}$

$\mathcal{A}_2 = \{x_0 - 1, x - 1, y\}$

$\mathcal{A}_3 = \{27x_0^2 - 94x_0 + 31, 14x - 9x_0 - 3, 7y^2 + 5x_0 - 3\}$

$\mathcal{A}_3 = 0$ 没有实解. 依据上面的定理知道, $K_1 = \emptyset$, $K_2 = \{1\}$. 因此有限核 $K = \{1\}$, 它是点 $(x_0, x, y) = (1, 1, 0)$ 到 x_0 上的投影, 所以唯一的极值 $f = 1$ 出现在唯一的极值点 $(x, y) = (1, 0)$.

本例调用 MMP 函数EVal的计算过程如下:

```
>h0:=x^2+y^2-x0;
>h1:=(x-1)^3+y^2;
>ord:=[y,x,x0];
>H:=[h0, h1];
>EVal(H, ord);
[[x^2+y^2-x0,x^3-4*x^2+3*x+x0-1], [y,x-1,x0-1], [7*y^2+5*x0-3,
14*x-9*x0-3, 27*x0^2-94*x0+31]]
```

2. 不等式的自动证明. 我们将利用上面的方法来证明有关三角形的不等式.

例 8.2.5 证明不等式

$$\sin\frac{A}{2} + \sin\frac{B}{2} + \sin\frac{C}{2} \leqslant \frac{3}{2}$$

首先我们引进新变元 $x_1, x_2, y_1, y_2, z_1, z_2$. 令 $x_1 = \sin\frac{A}{2}, x_2 = \cos\frac{A}{2}, y_1 = \sin\frac{B}{2}, y_2 = \cos\frac{B}{2}, z_1 = \sin\frac{C}{2}, z_2 = \cos\frac{C}{2}C$, 则有 $x_1^2 + x_2^2 = 1, y_1^2 + y_2^2 = 1, z_1^2 + z_2^2 = 1$.

既然 A, B, C 是三角形的三个角, 则我们有

$$A + B + C = \pi$$

因此有

$$\sin\frac{A+B}{2} = \sin\left(\frac{\pi}{2} - C\right) = \cos\frac{C}{2}$$

也就是

$$\sin\frac{A}{2}\cos\frac{B}{2} + \cos\frac{A}{2}\sin\frac{B}{2} = \cos\frac{C}{2}$$

同样地

$$\cos\frac{A+B}{2} = \cos\left(\frac{\pi}{2} - C\right) = \sin\frac{C}{2}$$

即有

$$\cos\frac{A}{2}\cos\frac{B}{2} - \sin\frac{A}{2}\sin\frac{B}{2} = \sin\frac{C}{2}$$

这样, 三角形式转化为了关于变元 $x_1, x_2, y_1, y_2, z_1, z_2$ 多项式的形式:

$$x_1y_2 + x_2y_1 = z_2, \quad x_2y_2 - x_1y_1 = z_1$$

形成多项式组 $\mathbb{P} = \{P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9\}$, 其中

$$\begin{aligned}
P_1 &= x_1^2 + x_2^2 - 1\\
P_2 &= y_1^2 + y_2^2 - 1\\
P_3 &= z_1^2 + z_2^2 - 1\\
P_4 &= x_1y_2 + x_2y_1 - z_2\\
P_5 &= x_2y_2 - x_1y_1 - z_1
\end{aligned}$$

要证明的结论

$$\sin\frac{A}{2} + \sin\frac{B}{2} + \sin\frac{C}{2} \leqslant \frac{3}{2}$$

变为了

$$x_1 + y_1 + z_1 \leqslant \frac{3}{2}$$

再引进新变元 r, 令 $P = x_1 + y_1 + z_1 - r$, 设

$$\mathbb{P}' = \mathbb{P} \cup \{P\} = \{P_1, P_2, P_3, P_4, P_5, P\}$$

这个问题的极大值总是存在的. 对变元序 $r < x_1 < y_1 < z_1 < x_2 < y_2 < z_2$, 令 $D = x_1y_1z_1(x_1+1)(y_1+1)(z_1+1)$, 得到 $\mathbb{P}'$ 的零点分解:

$$Zero(\mathbb{P}'/D) = \cup_{i=1}^{4}\mathrm{Zero}(\mathcal{A}_i/I_{\mathcal{A}_i}S_{\mathcal{A}_i}D)$$

其中

$$\begin{aligned}
\mathcal{A}_1 = & \{-2x_1^2y_1 - 2x_1y_1^2 + 2x_1y_1r + 2x_1^2 + 2x_1y_1 - 2x_1r + 2y_1^2 - 2y_1r + r^2 - 1, \\
& x_1 + y_1 + z_1 - r, x_1^2 + x_2^2 - 1, \\
& -x_2y_2 + x_1y_1 - x_1 - y_1 + r, -x_2z_2 - x_1^2 - x_1y_1 + x_1r + y_1\} \\
\mathcal{A}_2 = & \{r - 1, -1 + x_1, y_1 + z_1, x_2, y_1^2 + y_2^2 - 1, -z_2 + y_2\} \\
\mathcal{A}_3 = & \{x_1^2 + 2x_1 - 2x_1r + r^2 - 2, -x_1 - 2y_1 + r, \\
& -x_1 - 2z_1 + r, -x_2^2 - 1 + 2x_1 - 2x_1r + r^2 \\
& -2x_2y_2 - x_1r + x_1 + r + r^2 - 2, -2x_2z_2 - x_1r + x_1 + r + r^2 - 2\} \\
\mathcal{A}_4 = & \{-3 + 2r, 2x_1 - 1, 2y_1 - 1, 2z_1 - 1, 4x_2^2 - 3, 4x_2y_2 - 3, 4x_2z_2 - 3\}
\end{aligned}$$

从上面形式的零点分解, 我们可以得到一个有限核：

$$K = \cup_{i=1}^{4} K_i = \left\{1, \frac{3}{2}\right\}$$

从有限核定理, 我们知道

$$x_1 + y_1 + z_1 \leqslant \frac{3}{2}$$

即

$$\sin\frac{A}{2} + \sin\frac{B}{2} + \sin\frac{C}{2} \leqslant \frac{3}{2}$$

而且上述的最大值在分支 $\mathcal{A}_4$ 中取到. 从 $\mathcal{A}_4 = 0$, 容易知道 $r = \frac{3}{2}$, 而且 $x_1 = y_1 = z_1 = \frac{1}{2}$, $x_2 = y_2 = z_2 = \frac{\sqrt{3}}{2}$. 此时, 三角形的三个角 A, B, C 有 $A = B = C = \frac{\pi}{3}$, 即三角形为正三角形时, 等式成立.

本例在 MMP 中的计算过程如下：

```
>ps:=[x1^2+x2^2-1, y1^2+y2^2-1, z1^2+z2^2-1,
x1*y2+x2*y1-z2,x2*y2-x1*y1-z1,x1+y1+z1-r];
>ord:=[z2, y2,x2, z1, y1,x1, r];
>EVal(ps, ord, [x1, y1, z1,x1+1, y1+1, z1+1]);
[[-x1^2-y1*x1+r*x1-z2*x2+y1, y1*x1-x1-y2*x2-y1+r,x1^2+x2^2-1,
-x1-y1-z1+r, -2*y1*x1^2+2*x1^2-2*y1^2*x1+2*r*y1*x1+2*y1*x1
-2*r*x1+2*y1^2-2*r*y1+r^2-1],
[y2-z2, y1^2+y2^2-1,x2, y1+z1,x1-1, r-1],
[-r*x1+x1-2*z2*x2+r^2+r-2, -r*x1+x1-2*y2*x2+r^2+r-2,
-2*r*x1+2*x1-x2^2+r^2-1,  -x1-2*z1+r,
-x1-2*y1+r,x1^2-2*r*x1+2*x1+r^2-2],
[4*z2*x2-3, 4*y2*x2-3, 4*x2^2-3, 2*z1-1, 2*y1-1, 2*x1-1, 2*r-3]]
```

§8.3 方程实根个数的判定

方程组实根个数的判定. 考虑如下问题：求参数 $\mathbb{U}=(u_1,\cdots,u_d)$ 必须满足的充要条件, 以使如下的多项式方程组对变元 $\mathbb{X}=(x_1,\cdots,x_s)$ 有正好 n 个相异实解：

$$\mathbb{P}:\begin{cases} h_1(\mathbb{U},\mathbb{X})=0,h_2(\mathbb{U},\mathbb{X})=0,\cdots,h_s(\mathbb{U},\mathbb{X})=0\\ g_1(\mathbb{U},\mathbb{X})\geqslant 0,g_2(\mathbb{U},\mathbb{X})\geqslant 0,\cdots,g_t(\mathbb{U},\mathbb{X})\geqslant 0\end{cases}$$

其中

$$h_i,g_j\in\mathbb{Q}(u_1,\cdots,u_d)[x_1,\cdots,x_s],1\leqslant i\leqslant s,1\leqslant j\leqslant t$$

另外, 我们总假设 $\{h_1(\mathbb{U},\mathbb{X})=0,h_2(\mathbb{U},\mathbb{X})=0,\cdots,h_s(\mathbb{U},\mathbb{X})=0\}$ 对 $\mathbb{X}$ 仅有有限多个解.

首先, 由吴特征列方法, 可以将 $h_1(u,x)=0$, $h_2(u,x)=0,\cdots,h_s(u,x)=0$ 化为有限个升列形式. 然后, 如果必要的话, 将这些三角列分解为正则升列且关于每个 $g_j,j=1,\cdots,t$ 都是可逆的. 这样, 我们只需考虑如下情形.

求参数 $\mathbb{U}$ 所满足的充要条件, 以使得如下系统 $\mathbb{T}$ 对 $\mathbb{X}$ 恰有 $n(\geqslant 0)$ 个不同实解：

$$\mathbb{T}:\begin{cases} f_1(\mathbb{U},x_1)=0\\ f_2(\mathbb{U},x_1,x_2)=0\\ \quad\cdots\cdots\cdots\cdots\\ f_s(\mathbb{U},x_1,x_2,\cdots,x_s)=0\\ g_1(\mathbb{U},\mathbb{X})\geqslant 0,g_2(\mathbb{U},\mathbb{X})\geqslant 0,\cdots,g_t(\mathbb{U},\mathbb{X})\geqslant 0\end{cases}\tag{8.5}$$

其中

$$\mathbb{U}=(u_1,u_2,\cdots,u_d),\ \mathbb{X}=(x_1,x_2,\cdots,x_s)$$

$$f_i\in\mathbb{Q}(\mathbb{U})[x_1,\cdots,x_i],\quad 1\leqslant i\leqslant s$$

$$g_j\in\mathbb{Q}(\mathbb{U})[\mathbb{X}],\quad 1\leqslant j\leqslant t$$

$\mathcal{A}:f_1,f_2,\cdots,f_s$ 对变元序 $x_1<\cdots<x_s$ 是正则升列. (8.5) 中的某些不等式可以是严格的.

通过将一般形式的多项式集合转变为升列形式, 多项式组的零点个数问题在一定意义下变为单个多项式的零点个数问题. 而这一问题可以用判别式序列解决.

给定多项式 $g\in\mathbb{K}[\mathbb{X}]$ 和变元 x, g' 是 g 关于变元 x 的导数, 我们称 $\mathrm{Res}(g,g',x)$, 为 g 关于 x 的*判别式*, 记为 $\mathrm{Discrim}(g,x)$ 或在意义清楚时简记为 $\mathrm{Discrim}(g)$.

给定符号系数多项式

$$f(x) = a_0x^n + a_1x^{n-1} + \cdots + a_n$$

其系数构成的 $2n \times 2n$ 阶矩阵

$$\begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_n & & & & \\ 0 & na_0 & (n-1)a_1 & \cdots & a_{n-1} & & & & \\ & a_0 & a_1 & \cdots & a_{n-1} & a_n & & & \\ & 0 & na_0 & \cdots & 2a_{n-2} & a_{n-1} & & & \\ & & & \vdots & \vdots & \vdots & & & \\ & & & a_0 & a_1 & a_2 & \cdots & a_n \\ & & & 0 & na_0 & (n-1)a_1 & \cdots & a_{n-1} \end{bmatrix}$$

称作 $f(x)$ 的*判别矩阵*, 它的行列式的值记为 $\mathrm{Discr}\,(f)$. 用 d_k 或 $d_k(f)$ $(k = 1, 2, \cdots, 2n)$ 记由 $\mathrm{Discr}\,(f)$ 的前 k 行 k 列构成的子矩阵的行列式.

令 $D_0 = 1, D_k = d_{2k}, k = 1, \cdots, n$, 称

$$[D_0,\ D_1,\ D_2, \cdots, D_n]$$

为 $f(x)$ 的*判别式序列*, 记为 $\mathrm{DiscrList}\,(f)$. 显然, D_n 就是 $\mathrm{Discrim}(f, x)$.

称

$$[\mathrm{sign}(A_0), \mathrm{sign}(A_1), \mathrm{sign}(A_2), \cdots, \mathrm{sign}(A_n)]$$

为给定序列 $A_0, A_1, A_2, \cdots, A_n$ 的*符号表*, 其中

$$\mathrm{sign}(x) = \begin{cases} 1, & x > 0, \\ 0, & x = 0, \\ -1, & x < 0 \end{cases}$$

给定符号表 $[s_1, s_2, \cdots, s_n]$, 其*符号修订表*

$$[t_1, t_2, \cdots, t_n]$$

按如下规则构造:

- 如果 $[s_i, s_{i+1}, \cdots, s_{i+j}]$ 是所给符号表中的一段, 并且

$$s_i \neq 0, s_{i+1} = \cdots = s_{i+j-1} = 0, s_{i+j} \neq 0$$

则将此段中由 0 构成的序列

$$[s_{i+1}, \cdots, s_{i+j-1}]$$

替换为序列 $[-s_i, -s_i, s_i, s_i, -s_i, -s_i, s_i, s_i, \cdots]$ 中的前 $j-1$ 个, 也就是令

$$t_{i+r} = (-1)^{[(r+1)/2]} \cdot s_i, \quad r = 1, 2, \cdots, j-1$$

- 除此之外, 令 $t_k = s_k$, 即其余各项保持不变.

由判别式序列可以给出单变量方程实根的个数：

定理 8.3.1 ([105])　*给定实系数多项式*

$$f(x) = a_0x^n + a_1x^{n-1} + \cdots + a_n$$

如果判别式序列

$$[D_0, D_1(f), D_2(f), \cdots, D_n(f)]$$

的符号修订表的变号数是 ν, 则 $f(x)$ 的互异共轭虚根对的数目是 ν. 而且, 如果该符号修订表中非零元的个数是 l, 则 $f(x)$ 的相异实根的数目是 $l-1-2\nu$.

结合定理 8.3.1 与吴特征列方法即可解决本节开始所提的问题. MMP 的函数 tofind(fs, gs, ns, ds, vars, pvars, n) 即根据上述思想编制, 其中

- fs=$\{f_1, \cdots, f_t\}$ 是关于参数和变元的多项式列表.
- gs=$\{g_1 \cdots, g_s\}$ 是所有 “$g_i \geqslant 0$” 的约束条件序列.
- ns=$\{n_1 \cdots, n_k\}$ 是所有 “$n_i > 0$” 的约束条件序列.
- ds=$\{d_1 \cdots, d_l\}$ 是所有 “$d_i \neq 0$” 的约束条件序列.
- vars 是主变元列表.
- pvars 是参变元列表.
- n 是一个非负整数或者一个范围 ([1, n]).

tofind 函数给出方程组 fs=0 在条件

$$g_1 \geqslant 0 \wedge \cdots \wedge g_s \geqslant 0 \wedge n_1 > 0 \wedge \cdots \wedge n_k > 0 \wedge d_1 \neq 0 \wedge \cdots \wedge d_l \neq 0$$

下有 n 个实根的条件.

这一命令给出在满足所有约束条件下, 参数满足什么条件时, 方程组 fs 的根的个数在 [1,n] 的范围内. 函数如果运行成功, 返回一个列表. 注意, 程序计算过程不考虑边界, 即只考虑维数最高的情形. 以下提供几个例子.

例 8.3.2　*我们知道以 a, b, c 为边长的三角形存在的充分必要条件是 $a+b>c, a+c>b, b+c>a$. 试求以一条边长为 a 及该边上的高为 h_a 和外接圆半径为 R 的三角形存在的充分必要条件.*

```
>tofind([a^2*ha^2-4*s*(s-a)*(s-b)*(s-c), 2*R*ha-b*c, 2*s-a-b-c], [],
[a, b, c, a+b-c, b+c-a, c+a-b, R, ha], [], [s, b, c], [a, R, ha],
[1, n]);
```

```
The system has required real solution IF AND ONLY IF
    [R1<0, R3>0]
OR
    [R1<0, R2<0, R3<0]
where
    R1==a-2*R
    R2==a^2+4*ha^2-8*R*ha
    R3==a^2-4*R*ha
PROVIDED THAT
    a-2*R!=0
    a+2*R!=0
    a^2+4*ha^2-8*R*ha!=0
    a^2+4*ha^2+8*R*ha!=0
Time is: 6.531 second(s)
```

例 8.3.3 s, r, ma 需要满足什么条件才可以构造一个三角形使得其半周长为 s, 内切圆半径为 r, 一条边上的中线长度为 ma?

```
>tofind([4*ma^2-2*(x+z)^2-2*(x+y)^2+(y+z)^2, s-x-y-z,
r^2*(x+y+z)-x*y*z], [], [x, y, z, s, r, ma], [], [x, y, z],
[s, r, ma], [1, n]);
The system has required real solution IF AND ONLY IF
  [R1<0]
where
  R1==ma^3-s^2*ma+2*r*s^2
PROVIDED THAT
  -ma^3+s^2*ma+2*r*s^2!=0
  ma^3-s^2*ma+2*r*s^2!=0
  -16*ma^6+8*s^2*ma^4-s^4*ma^2-36*r^2*s^2*ma^2+r^2*s^4+27*r^4*s^2!=0
Time is: 6.485 second(s)
```

§8.4 优化问题的数值计算与随机搜索方法

前面介绍的几种算法都是针对代数系统的, 而且都是精确算法. 但实际问题中往往会遇到求超越函数极值问题. 此时前面介绍的精确方法已经不再适用. 一般讲, 最优化方法解决的数学规划问题主要有两类：线性规划和非线性规划. 一般情况下,

利用单纯型法可以解出线性规划问题的最优解. 对于非线性规划问题, 则有一些比较成熟的算法进行求解, 但是大部分算法都是求非线性规划的局部最优解, 而要确定非线性规划问题的全局最优解则是非常困难的. 本章将介绍怎样用 MMP 进行最优化问题的求解.

8.4.1 线性规划

线性规划是由目标函数为决策变量的线性函数和约束条件为线性等式或不等式组成的数学规划, 它是数学规划中较简单的一类, 因而也是最基本的一类数学规划问题.

一般的线性规划问题, 总可以化为如下的标准形式.

$$\begin{cases} \min \quad c^T x \\ \text{s.t.} \quad Ax = b, x \geqslant 0 \end{cases} \tag{8.6}$$

其中, $A = (a_{ij})_{m\times n}, x = (x_1, x_2, \cdots, x_n)^T, \text{rank}(A) = m \leqslant n, c = (c_1, c_2, \cdots, c_n)^T$, $b = (b_1, b_2, \cdots, b_n)^T, b \geqslant 0$.

1. 线性规划的单纯型法. 单纯型法是求解线性规划问题的一种通用的有效算法, 由 Dantzig 于 1947 年提出. 一个线性规划若有最优解, 则一定有最优的基本可行解, 而且基本可行解的个数是有限的. 因此, 一个求线性规划的直观想法是, 把所有的基本可行解求出来, 并求出相应的目标函数值. 比较所求得的目标函数值, 就可求得其中目标函数值的最优解了. 单纯型算法的基本思想是给出一种规则, 使得从线性规划问题的一个基本可行解转移到另一个基本可行解时目标函数值是减小的, 而且两个基本可行解之间的转换是容易实现的. 经过有限次迭代, 即可求得所需的最优基本可行解.

MMP 中实现了单纯型算法和二阶段单纯型算法, 其语法说明和计算实例如下所示.

标准单纯型法语法：

BasicSimplexSolve(极大极小标识, 目标函数, 约束条件表, 决策变量表, 基本变量表, 非基本变量表)；

极大极小标识只能取值 “max” 或 “min”. 取值为 “max” 时表示对目标函数求最大值, 否则求最小值.

下面给出两个例子：

```
>f:=5*x1+3*x2;
>ps:=[x1+x2+x3-3500,
     x1+x4-1500,
```

```
      5*x1+2*x2+x5-10000];
>BasicSimplexSolve("max",f,ps,[x1,x2,x3,x4,x5],[0,1,3],[2,4]);
The optimal solution vector is: [1000,2500,0,500,0]
The optimal function value is: 12500
>f:=4*x1+x2+x3;
>ps:=[2*x1+x2+2*x3-4,3*x1+3*x2+x3-3];
>BasicSimplexSolve("min",f,ps,[x1,x2,x3],[0,2],[1]);
The optimal solution vector is: [0,0.4,1.8]
The optimal functionvalue is: 2.2
```

MMP 还实现了二阶段单纯型法. 调用方法如下：
GeneralSimplexSolve(极大极小标识, 目标函数, 约束条件表, 决策变量表, 变量下限, 变量上限);
极大极小标识只能取值 “max” 或 “min”. 取值为 “max” 时表示对目标函数求最大值, 否则求最小值.

下面给出两个例子：

```
>f:=2*x1+x2+3*x3-2*x4+10*x5;
>ps:=[x1+x3-x4+2*x5-8,x2+2*x3+2*x4+x5-13];
>GeneralSimplexSolve("max",f,ps,[x1,x2,x3,x4,x5],[],[7,10,1,5,3]);
The optimal solution vector is: [2,10,0,0,3]
The optimal function value is: 44
>f:= 2*x1+x2+x3;
>ps:=[4*x1+6*x2+3*x3-8,
      x1-9*x2+x3+3,
      -2*x1-3*x2+5*x3+4];
>GeneralSimplexSolve("min",f,ps,[x1,x2,x3],[],[]);
The optimal solution vector is: [0,1.33333,0]
The optimal function value is: 1.33333
```

2. 线性多目标规划的目标规划方法. 在许多实际问题中, 衡量一个设计方案的好坏标准往往不只一个. 例如：设计一个导弹, 既要射程最远, 又要燃料最少还要重量最轻, 精度最高. 在把飞机外形设计包括在内的飞机最优设计问题中, 不仅要求飞机的总重量最轻, 还要求在耗油量一定的情况下, 航程最远. 在确定一个橡胶配方时, 往往要同时考察强力、硬度、变形、伸长等多个指标. 这一类问题称为多目标数学规划问题或多目标最优化问题, 它们的早期来源是经济理论, 现在已应用于工程

优化设计、地区发展规划、数理经济学和环境保护问题等许多领域.

多目标规划问题的一般形式为

$$\begin{cases} V-\min & F(x)=(f_1(x),f_2(x),\cdots,f_p(x))^T \\ \text{s.t.} & g_i(x)\leqslant 0, i=1,2,\cdots,m \end{cases} \tag{8.7}$$

其中, $x=(x_1,x_2,\cdots,x_n)^T, p\geqslant 2$.

多目标规划是由线性规划发展而来, 线性规划归根到底是研究资源的有效分配和利用. 模型的特点是, 在满足一组约束条件的情况下, 寻求某一目标 (如产量、利润、成本) 的最大值和最小值. 线性规划只能解决一个目标优化的问题. 而在实际中, 更多的是要考虑多个目标的问题, 这些目标往往具有一定的制约性、冲突性, 甚至还具有矛盾性, 为了较好的处理这类问题, 在线性规划基础上产生了目标规划方法.

目标规划是解决多目标规划的一种有效方法, 基本思想是将实际问题要考虑的多个目标按重要性递减排序, 再结合实际情况给每个目标赋予实际值, 然后寻求一种方案, 使之尽可能满足这些目标. 若这些目标不能同时满足, 则优先满足比较重要的目标. 对于不能满足的目标, 目标规划可以给出不能实现的程度, 以供决策者参考.

MMP 中实现了目标规划算法, 其语法说明和计算实例如下所示.

LnMultiGoalSolve(极大极小标识, 目标函数, 决策变量表, 变量下限, 变量上限, 优先级变量表, 正偏差变量表, 负偏差变量表);

极大极小标识只能取值 “max” 或 “min”. 取值为 “max” 时表示对目标函数求最大值, 否则求最小值.

下面给出两个例子：

```
>f:=L1*n1+L2*p4+L3*(5*n2+3*n3)+L4*p1;
>ps:=[x1+x2+n1-p1-80,
      x1+n2-p2-70,
      x2+n3-p3-45,
      n4+p1-p4-10];
>LnMultiGoalSolve("min",f,ps,[x1,x2],[],[],[L1,L2,L3,L4],
                  [p1,p2,p3,p4],[n1,n2,n3,n4]);
The Optimal Solution Vector Is: [70,20]
The Optimal Goal Funtion Value Is: [90,70,20,10]
The Optimal Diff Vector Is: [0,0,75,10]
The Optimal Positive Diff Vector Is: [10,0,0,0]
```

```
The Optimal Negetive Diff Vector Is: [0,0,25,0]
>f:=L1*(p1+p2)+L2*n3+L3*p4;
>ps:=[2*x1+x2+n1-p1-12,
      x1+x2+n2-p2-10,
      x1+n3-p3-7,
      x1+4*x2+n4-p4-14];
>LnMultiGoalSolve("min",f,ps,[x1,x2],[],[],[L1,L2,L3],
                  [p1,p2,p3,p4],[n1,n2,n3,n4]);
The Optimal Solution Vector Is: [6,0]
The Optimal Goal Funtion Value Is: [12,6,6,6]
The Optimal Diff Vector Is: [0,1,0]
The Optimal Positive Diff Vector Is: [0,0,0,0]
The Optimal Negetive Diff Vector Is: [0,4,1,8]
```

8.4.2　非线性规划问题的局部优化

非线性规划的实质是求任意一个多元函数 (目标函数) 在给定域 (约束条件) 上的最大值或最小值. 如果在约束条件或目标函数中至少有一个是非线性函数, 这样的问题就称为非线性规划问题. 非线性规划的基本定理是 Kuhn 和 Tucker 于 1951 年提出的. 之后对于非线性规划的最优性条件、对偶性和稳定性方面的研究, 以及对各种类型的非线性规划问题提出了各种算法, 使非线性规划在理论上和计算方法方面逐渐完善. 特别是由于非线性规划问题对于目标函数和约束条件几乎没有任何限制, 使得非线性规划越来越广泛地应用于最优设计、经济管理等各个领域. 但是, 各种非线性规划的计算方法都有自己的特定适用范围, 到目前为止, 还没有适用于各种问题的一般计算方法.

一般地, 非线性规划问题的标准形式记为

$$\begin{cases} \min \quad f(x) \\ \text{s.t.} \quad g_i(x) \geqslant 0, i=1,2,\cdots,m \end{cases} \tag{8.8}$$

其中, $x=(x_1,x_2,\cdots,x_n)$, 目标函数 $f(x)$ 和约束条件 $g_i(x)(i=1,2,\cdots,m)$ 中至少有一个是非线性函数.

3. 一维最优化方法.　在求解无约束非线性规划问题:

$$\min \quad f(x), x\in\mathbb{R}$$

的最优解时, 往往采用迭代方法: 首先求得一个初始点 $x^1\in\mathbb{R}$, 然后按一定方法求得一个方向 $d^1\in\mathbb{E}^n$. 并从 x^1 出发, 沿方向 d^1 求 $f(x)$ 的最优解. 即求单变量极值

问题：

$$\min_{\lambda} f(x^1+\lambda d^1)=f(x^1+\lambda_1 d^1)$$

并使 $x^1+\lambda_1 d^1\in\mathbb{R}$, 这样就得到 $x^2=x^1+\lambda_1 d^1$.

一般地, 若已求得点 x^k, 应立即求方向 $d^k\in\mathbb{E}^n$, 并求解单变量极值问题

$$\min_{\lambda} f(x^k+\lambda d^k)=f(x^k+\lambda_k d^k)$$

并使 $x^k+\lambda_k d^k\in\mathbb{R}$, 令 $x^{k+1}=x^k+\lambda_k d^k$, 如此进行下去就可以产生一个点列 $\{x^k\}$, 逐渐逼近问题的最优解. 这种产生点列的规则就称为算法. 非线性规划的各种算法的差别, 就在于确定搜索方向 d^k 的方法是不同的. 同时, 在各种算法中往往要多次求解单变量函数的极值问题. 这类问题也称一维最优化问题.

MMP 中实现了一维极值问题的黄金分割算法, 其语法说明和计算实例如下所示.

GoldenDiv(极大极小标识, 目标函数, 变量表, 变量下限, 变量上限);

极大极小标识只能取值 “max” 或 “min”. 取值为 “max” 时表示对目标函数求最大值, 否则求最小值.

下面给出两个例子：

```
>f:=0.2*x^5-1.6666666*x^3+4*x;
>GoldenDiv("min",f,[x],[-1.854],[ 2.292]);
The optimal solution is: -1
The optimal function value is: -2.53333
>f:=x^2-2*x+1;
>GoldenDiv("min",f,[x],[-1],[2]);
The optimal solution is: 1
The optimal function value is: 0
```

4. 无约束最优化方法. 无约束极值问题可表述为

$$\min\quad f(x), x\in\mathbb{R}^n$$

在求解上述问题时常使用迭代法. 迭代法大体分为两类：一类要用到函数的一阶导数和二阶导数, 由于用到了函数的解析性质, 故称为 *解析法*；另一类在迭代过程中仅用到函数数值, 而不要求函数的解析性质, 这种方法称为 *直接法*. 一般说来直接法的收敛速度较慢, 只有在变量较少时才适用. 但直接法的迭代步骤简单, 特别是当目标函数的解析式十分复杂, 甚至写不出具体的表达式时, 它们的导数很难求解, 或根本不存在. 这时, 就只有用直接法了.

求多个变量的无约束极小值的基本下降算法主要有：最速下降法、Newton 法和阻尼 Newton 法. 这些算法为我们提供了一些简单而直接的求解方案, 更重要的也许是它们为求解无约束极值问题的难度和收敛速度提供了可供进行比较的标准, 为建立更有效的算法奠定了基础.

最速下降法, 也叫 *梯度法*, 是人们用来求多变量函数极值问题的最早的一种方法. 后来提出的不少方法都是人们企图改进这种算法的结果. *Newton法* 是求解一维问题的 Newton 法的推广, 这个方法也是求解无约束极值问题的最古老的算法之一, 已发展为一类应用广泛的重要算法 — 拟 Newton 型算法. *阻尼Newton法* 保持了 Newton 法快速收敛的优点, 又不要求初始点选的很好, 因而在实际应用中取得了较好的效果.

前面讲过的最速下降法, 计算步骤简单, 但收敛速度太慢, 而 Newton 法和阻尼 Newton 法收敛速度快, 但要计算二阶偏导数矩阵及其逆阵, 计算量太大. 因此人们希望找到一种方法, 它兼有这两种方法的优点, 又能克服它们的缺点. *共轭方向法* 就是这样的一类方法, 它比最速下降法的收敛速度要快的多, 同时又避免了像 Newton 法所要求的 Hessian 矩阵的计算、存储和求逆.

共轭方向法, 主要是其中的共轭梯度法, 它对一般目标函数的无约束优化问题的求解具有较高的效率, 因此在无约束优化算法中占有重要的地位, 是目前最常用的方法之一, 由于它的计算公式简单, 存储量少, 可以用来求解比较大的问题, 特别是用于最优控制问题时, 效果很好, 引起了人们的重视和兴趣.

变尺度法, 是由 W.C.Davidon 于 1959 年首先提出, 1963 年为 R.Fletcher 和 M.J.D.Powell 所发展和修正, 形成为著名的 DFP 变尺度算法, 目前变尺度法已发展为一大类算法, 被认为是求解无约束优化问题最有效的算法之一, 得到了广泛的研究和应用.

上面讲的几种方法, 都要利用目标函数的一阶或二阶偏导数. 但在实际问题中, 所遇到的目标函数往往比较复杂, 有的甚至难于写出其明显的解析表达式, 因此, 它们的导数很难求得, 甚至根本无法求得, 这时, 就必须采用不用导数的方法, 即求多变量函数极值的*直接搜索法*, 这类方法的特点是方法简单, 适用范围较广, 但由于没有利用函数的分析性质, 其收敛速度一般是比较慢的.

MMP 中实现了直接搜索算法, 其语法说明和计算实例如下所示.

DirectSearch(极大极小标识, 目标函数, 变量表);

极大极小标识只能取值 “max” 或 “min”. 取值为 “max” 时表示对目标函数求最大值, 否则求最小值.

下面给出两个例子：

```
>f:=x1^2+6*x1+x2^2+8*x2-20;
```

```
>DirectSearch("min",f, [x1,x2]);
The optimal solution vector is: [-3,-4]
The optimal function value is: -45
>f:=x1^2-2*x1+x2^2-3*x2+2;
>DirectSearch("min",f,[x1,x2]);
The optimal solution vector is: [1,1.5]
The optimal function value is: -1.25
```

5. 约束最优化方法. 一般的约束优化问题是指：

$$\begin{array}{ll} \min & f(x) \\ \text{s.t} & g_i(x) \leqslant 0, i=1,2,\cdots,m \\ & h_j(x)=0, j=1,2,\cdots,p \\ & x \in X \subset \mathbb{R}^n \end{array} \tag{8.9}$$

其中, $x=(x_1,x_2,\cdots,x_n)^T$.

约束优化问题是应用中经常遇到的一类数学规划问题, 它的解法是人们最为关心的, 因此研究的人比较多, 提供的方法也比较多, 但目前尚没有一种对一切问题都普遍有效的算法, 而且求得的解多是局部最优解.

约束优化方法大体可以分为以下四类：

1. 用线性规划或二次规划来逐次逼近非线性规划的方法, 如 SLP 法、SQP 法等；
2. 把具有约束的优化问题转化为无约束优化问题来求解的方法, 如 SUMT 外点法、SUMT 内点法等；
3. 对约束优化问题不预先作转换, 直接进行处理的分析方法, 如可行方向法、梯度投影法、既约梯度法等；
4. 对约束优化问题不预先作转换的直接求解方法, 如复形法、随机实验法等.

惩罚函数法 和 *碰壁函数法* 是早期求解约束优化问题的一类重要而常用的方法, 其基本思想是把约束问题转化为一个或一系列无约束问题来求解, 所以也称为无约束极小化技术, 简称为 SUMT 法, 通常把惩罚函数法称为 SUMT 外点法, 碰壁函数法称为 SUMT 内点法.

1960 年, Rosen 针对线性约束优化问题首先提出了 *梯度投影法*, 1961 年 Rosen 又将他的方法推广到处理非线性约束的情况, 后来这类方法又得到了进一步的发展, 成为求解非线性规划问题的一类重要的方法.

MMP 中实现了惩罚函数法、碰壁函数法以及梯度投影法, 其语法说明和计算实例如下所示.

InnerPointSUMTMax(极大极小标识, 目标函数, 约束条件表, 变量表, 变量初值表)；

极大极小标识只能取值 “max” 或 “min”. 取值为 “max” 时表示对目标函数求最大值, 否则求最小值.

OuterPointSUMTMax(极大极小标识, 目标函数, 约束条件表, 变量表, 变量初值表, 等式约束个数);

极大极小标识只能取值 “max” 或 “min”. 取值为 “max” 时表示对目标函数求最大值, 否则求最小值.

GradientProject(极大极小标识, 目标函数, 约束条件表, 变量表, 变量初值表);

极大极小标识只能取值 “max” 或 “min”. 取值为 “max” 时表示对目标函数求最大值, 否则求最小值.

下面给出三个例子：

```
>f:=(x1-3)^2+(x2-2)^2+4;
>ps:=[x1+x2-4];
>InnerPointSUMT("min", f,ps, [x1,x2],[1,1]);
The optimal solution vector is: [2.95292,1.04708]
The optimal function value is: 4.91027
>f:=(x1-3)^2+(x2-2)^2+4;
>ps:=[x1+x2-5,x1-x2-1];
>OuterPointSUMT("min",f,ps,[x1,x2],[1,1],1);
The optimal solution vector is: [3,2]
The optimal function value is: 4
>f:=(x1-3)^2+(x2-2)^2+4;
>ps:=[-x1-x2+2,-x1-5*x2+5,x1,x2];
>GradientProject("min",f,ps,[x1,x2],[1,1]);
The optimal solution vector is: [1.5,0.5]
The optimal function value is: 8.5
```

8.4.3　基于遗传算法的全局优化

演化算法 ([50]) 是一种模拟生物进化与机制求解问题的自组织、自适应人工智能技术. 这类技术的核心思想源于这样的基本认识：生物进化过程本身是一个自然的、并行发生的、稳健的优化过程. 这一优化过程的目标是对环境的自适应性, 生物种群通过 “适者生存”、“优胜劣汰” 和 “遗传变异” 来达到进化 (优化) 的目的. 适者生存理论消除了问题解中不适应的因素. 随机交换理论利用了原有解中的已有知识, 从而加速了对优化解的搜索过程. 依 Darwin 的自然选择与 Mendel 的遗传变异理论, 生物进化是通过繁殖、变异、竞争和选择这四种基本形式实现的. 因而, 如果

把待解决的问题理解为对某个目标函数的全局优化, 则演化算法就是建立在模拟上述生物进化过程基础上的一种基于群体的随机搜索技术.

演化算法包括遗传算法、进化规划、进化策略与遗传编程等. 遗传算法是 20 世纪 70 年代由美国的 Holland 提出的模仿生物进化过程的优化方法, 与 Fogel 提出的进化规划和 Rechenberg 与 Schwefel 提出的进化策略共同构成了演化计算领域的三大分支.

6. 遗传算法. 作为一种随机优化与搜索方法, 遗传算法具有如下特点：

- 遗传算法的操作对象是一组可行解, 而非单个可行解; 搜索轨道有多条, 而非单条, 因而具有良好的并行性.
- 遗传算法只需利用目标函数的取值信息, 无须梯度等高阶信息, 因而适用于大规模、非线性不连续多峰函数、无解析表达式或病态函数的优化.
- 遗传算法的选择机制是一种 "软" 决策, 使它具有良好全局优化性能和鲁棒性.
- 遗传算法操作的可行解集是经过编码的, 目标函数可解释为编码个体的适应值, 因而具有良好的可操作性与简单性.

由于遗传算法的上述普适性, 我们在 MMP 中实现了基于遗传算法的全局优化算法. 需要说明的是遗传算法是一种随机搜索算法, 所得到的结果一般并不能保证是真正的最优值.

遗传算法基本流程如下：

```
初始化群体 P(0) 和循环代数 t = 0;
评价初始群体 P(0);
while (不满足结束条件)
    t ：= t+1;
    选择运算：由上一代群体 P(t-1) 选择下一代群体 P(t);
    杂交运算;
    变异运算;
    评价群体 P(t);
end while
输出结果
```

下面简要解释上述算法中的基本概念与 MMP 中实现的遗传算法.

群体与编码. 上面已经提到过, 遗传算法的实际上是一种群体随机搜索算法. 我们首先需要对搜索空间中的个体进行编码. 最基本的编码方法是二进制编码, 即用一个定长的二进制数表示搜索空间的个体. 确定个体的编码形式后, 我们需要确定群体的所含个体的个数. 遗传算法将对给定初始群体进行随机变换, 得到具有相同个数的新一代群体. 所以, 上述算法中的参数 t 也称为进化的代数. 遗传算法的

中止条件可以是某个规定的进化代数, 也可以是某种稳定性条件. 例如, 如果群体的进化不能进一步改进目标函数的结果则算法中止.

杂交运算. 杂交算子是遗传算法用来生成新个体的两个算子之一. 下面介绍单点交叉算子的操作步骤.

- 随机在区间 [0, 1] 产生一个小数 r;
- 如果 r 小于一个给定的阀值 Pc, 则对该对个体施加杂交操作：
  - ⋆ 假设 n 是个体的长度. 从 2 到 n-1 随机产生一个整数作为交叉点, 记为 pos.
  - ⋆ 相互交换交叉点 pos 之后的两个染色体的基因,

$$\mathbf{v_1} = (b_1, \cdots, \mathbf{b_{pos}}, b_{pos+1}, \cdots, b_n)$$
$$\mathbf{v_2} = (c_1, \cdots, \mathbf{c_{pos}}, c_{pos+1}, \cdots, c_n)$$

产生两个新染色体：

$$\mathbf{v_1'} = (b_1, \cdots, \mathbf{b_{pos}}, c_{pos+1}, \cdots, c_n)$$
$$\mathbf{v_2'} = (c_1, \cdots, \mathbf{c_{pos}}, b_{pos+1}, \cdots, b_n)$$

变异运算. 遗传算法中的所谓变异运算, 是指将个体的某些位上的数值进行随机变换产生新的个体. 从遗传运算过程中产生新个体的能力方面来说, 杂交运算是产生新个体的主要方法, 它决定了遗传算法的全局搜索能力; 而变异运算只是产生新个体的辅助方法, 但它也是必不可少的一个步骤, 引进变异算子可以改善遗传算法的局部搜索能力. 遗传算法使用杂交算子已经从全局的角度出发找到一些较好的个体编码结构, 它们已接近或有助于接近问题的最优解. 但仅使用杂交算子无法对搜索空间的细节进行局部搜索, 使用变异算子来调整个体编码中的部分基因值, 就可以从局部的角度出发使个体更加逼近最优解, 从而有助于提高遗传算法的局部搜索能力.

现介绍 MMP 中用到的非均匀变异算子. 假设由第 k 次到第 $k+1$ 次搜寻函数 $f(x)$ 在 $[a,b]$ 上的最优解, 则

$$x_{k+1} = \begin{cases} x_k + \Delta(t, b - x_k), \text{ 如果随机数 } \xi = 0 \\ x_k - \Delta(t, x_k - a), \text{ 如果随机数 } \xi = 1 \end{cases}$$

其中

$$\Delta(t, y) = y \cdot (1 - r^{(1-\frac{t}{T})^b})$$

r 是 [0, 1] 之间随机产生的均匀小数; T 是最大进化代数; b 是一个系统参数, 在产

生新实数时决定对循环代数的依赖程度. 函数 $\Delta(t,y)$ 返回一个 $[0,y]$ 之间的值, 且当循环变量 t 增大时, $\Delta(t,y)$ 趋近于 0. 该性质使得非均匀变异算子在程序开始阶段均匀搜索整个空间, 而在程序最后阶段仅在局部的区域进行搜索. 这种策略保证在程序后半段生成的后代比随机搜索要接近它的父代.

选择运算. MMP 采用所谓的联赛选择算子. 假设群体规模是 N, 联赛选择规模是 M(M $\leqslant$ N), 现介绍本算法用到的联赛选择算子, 即如何从生成的中间群体 X 选择出下一代群体解 XX：

```
for (i = 0; i < N; i++)
        XX[i] = X[i];
        for (j = 0; j < M; j++)
            从群体中随机选择一个个体 Y;
            如果 F(XX[i]) 优于 F(Y) 则 XX[i] = Y;
        End for;
End for.
```

7. 全局优化函数与示例. MMP 中基于遗传算法的优化函数格式如下：

```
gaoptimize(ps, vs, UB, LB);
gaoptimize(ps, vs, UB, LB, nTime,GENERATION, POPSIZE, TOURSIZE).
```

在以上命令中, 各参数的含义如下：ps 是目标函数的列表, vs 是方程中的变元列表, UB 是各变元搜索范围上界, LB 是各变元搜索范围下界. 第二条命令的后四个参数是遗传算法本身的参数, 其中 nTime：命令执行中遗传算法运行次数, 一般位于 [2, 30]; GENERATION：最大循环次数; POPSIZE：群体规模; TOURSIZE：联赛选择规模, 并且要求 POPSIZE⩾TOURSIZE. 若用户自己指定这四个参数, 可选用第二条命令；若用户不提供这四个参数, 可选用第一条命令, 此时系统将自动选择一组默认值：nTime = 5, GENERATION = 500, POPSIZE = 10, TOURSIZE = 4.

命令 gaoptimize 利用遗传算法, 求解目标函数 ps 在给定搜索范围 [LB, UB] 中的全局最优解. 因为演化算法是随机搜索算法, 所以每两次运行所得的结果一般来说不完全相同. gaoptimize 的输出结果是一个列表 [vals, minval], 其中 minval 是目标函数最优 (小) 值、vals 是相应变元的取值.

下面通过几个计算实例说明这两个优化函数的用法, 这些例子均见 [108].

Shubert 函数：$f=\sum\limits_{i=1}^{5} i\cos[(i+1)x_1+i]\times\sum\limits_{i=1}^{5} i\cos[(i+1)x_2+i],\ -10\leqslant x_i\leqslant 10.$
这个函数有七百多个局部极值, 其中十八个是全局最优值：$f_{\min}=-186.731$.

```
>gaoptimize([f], [x1,x2], [10, 10], [-10, -10], 5, 200, 10, 3);
```

```
[[-1.42513, 5.48286], -186.731]
```

输出结果表示函数在 $x_1=-1.42513, x_2=5.48286$ 处取到极小值 -186.731.

广义 Rastrigin 函数: $f=\sum_{i=1}^{5}[x_i^2-10\cos(2\pi x_i)+10]$, $-5.12\leqslant x_i\leqslant 5.12$. 这个函数的极小值是 $f_{\min}=0$, 全局最小值点是 $(0,\cdots,0)$.

```
>gaoptimize([f], [x0,x1,x2,x3,x4], [5.12,5.12,5.12,5.12,5.12],
   [-5.12, -5.12, -5.12, -5.12, -5.12], 5, 200, 10, 3);
[[-4.2245e-009, 2.1481e-008, 5.653e-009,
 -2.4252e-009, -3.0919e-009],1.00003e-013]
```

或者输入命令：

```
>gaoptimize([f], [x1,x2,x3,x4], [5, 5, 5, 5], [-5, -5, -5, -5]);
[[9.9113e-011,1.6031e-010,3.5341e-010,
 3.9055e-011,-1.9086e-011],1.62318e-019]
```

可以得到更好的最优值.

广义 Griewank 函数: $f=\frac{1}{4000}\sum_{i=1}^{5}x_i^2-\prod_{i=1}^{5}\cos(\frac{x_i}{i})+1$, $-600\leqslant x_i\leqslant 600$. 该函数的最优值是 $f_{\min}=0$, 全局最小值点是 $(0,\cdots,0)$.

```
>gaoptimize([f], [x1,x2,x3,x4,x5], [600, 600, 600, 600, 600],
           [-600, -600, -600, -600, -600], 10, 2000, 5, 3);
[[2.3874e-010,-2.1048e-010,1.4674e-010,
 -6.8057e-012,1.6764e-011],3.0791e-023]
```

8. 约束优化问题的实编码遗传算法. 目前, 实数编码的遗传算法在实际工程中使用较多, 由于没有编码和解码的资源开支, 对于复杂的优化问题, 计算速度与二进制编码相比具有明显优势. 实验证明, 通过设计专门的遗传算子, 在多数数值优化问题中, 实数编码的遗传算法与二进制编码相比具有较高的搜索效率.

MMP 中实现了处理无约束非线性规划和约束非线性规划的遗传算法, 其语法说明和计算实例如下所示.

语法:

max/min 目标函数;

GeneticAlgoSolve (终止类型, 变量表, 变量下限, 变量上限, 群体规模, 进化代数);

语法:

max/min 目标函数;

s.t.

约束条件 1,

······

约束条件 m;

GeneticAlgoSolve (终止类型, 变量表, 变量下限, 变量上限, 群体规模, 进化代数);

考虑下面例子:

```
>min 0.25*x^4-1.333*x^3+2.5*x^2-2*x;
>GeneticAlgoSolve("MAXGEN",[x],[0],[31],20,40);
The optimal solution vector is: [1.99599]
The optimal function value is: -0.664008
>min (x1-3)^2+(x2-2)^2;
s.t.
x1+x2-4;
>GeneticAlgoSolve("ADAPT",[x1,x2],[-3,-2],[20,20],100,150);
The optimal solution vector is: [3.17033,2.0642]
The optimal function value is: 0.0331348
>min x1^2+6*x1+x2^2+8*x2-20;
>GeneticAlgoSolve("MAXGEN",[x1,x2],[-10,-20],[30,40],100,200);
The optimal solution vector is: [-3,-4]
The optimal function value is: -45
```

文献说明

单变量多项式方程的实根隔离算法参见 [66]. MMP 中实现的多项式方程组实根隔离算法来自 [65]. 代数系统的有限核定理由吴文俊提出 ([87]). 用有限核定理证明三角函数不等式的工作来自 [77]. 代数方程组实根个数判定算法来自 [106]. MMP 中实现的遗传算法采用的非均匀变异算子来自 [108]. 关于非线性规划的数值方法请见 [10].

参 考 文 献

1 Ablowitz M J., Clarkson P A. Solitons, Nonlinear Evolution Equations and Inverse Scattering. Cambridge: Cambridge University Press, 1991

2 Aroca J M., Cano J., Feng R., Gao X S. Algebraic general solutions of algebraic ODEs. In: Proc. ISSAC'05. New York: ACM Press, 2005, 29–36

3 Aroca J M., Cano J., Jung F. Power series solutions for non-linear PDE's. In:Proc. ISSAC'03. New York: ACM Press, 2003, 15–22

4 Aubry P., Lazard D., Maza M M. On the theries of triangular sets. J. of Symb. Comput., 1999, 28: 105–124

5 Boulier F., Lazard D., Ollivier F., Petitiot M. Representation for the radical of a finitely generated differential ideal. In: Proc. ISSAC'95. New York: ACM Press, 1995, 158–166

6 Bouziane D., Rody A K., Maarouf H. Unmixed-dimensional decomposition of a finitely generated perfect differential ideal. J. of Symb. Comput., 2001, 31: 631–649

7 Bronstein M. Integration of elementary functions. J. of Symb. Comput., 1990, 9: 117–173

8 Briot C., Bouquet J. Propriétés des fonctionsdéfinies par des équations différentielles. Journalde l'Ecole Polytechnique, 1856, 36: 133–198

9 Buchberger B. Gröbner bases: an algorithmic method in polynomial ideal theory. In: Recent Trends in Multidimensional Systems Theory. D. Reidel Publ. Comp., 1985

10 Bazarra M S., Sherali H D., Shetty C M. Nonlinear Programming: Theory and Algorithms. John Wiley & Sons Inc., 1993

11 Cano J. The Newton polygon method for differential equations. In: H. Li, P. Olver , G. Sommer (eds.). LNCS 3519. 2005

12 Carnicer M M. The Poincaré problem in the nondicritical case. Ann. of Math., 1994, 140: 289–294

13 Chen X F., Wang D K. The projection of quasi variety and its application on geometric theorem proving and formula deduction. In: Automated Deduction in Geometry. Springer, 2004, 21–30

14 Chou S C., Yang J G. On the algebraic formulation of certain geometry statements and mechanical geometry theorem proving. Algorithmica, 1989, 4: 237–262

15 Chou S C. Mechanical Geometry Theorem Proving. Dordrecht, Netherlands: D. Reidel Publishing Company, 1988

16 Chou S C., Gao X S. A collection of geometry theorems proved mechanically using Wu's method, part on differential geometry. MM Preprints, 1991, 6

17 Chou S C., Gao X S. Automated reasoning in differential geometry and mechanics: part I, an improved version of Ritt-Wu's decomposition algorithm. J. of Automated Reasoning,1993, 10: 161–172

18 Chou S C., Gao X S. Automated reasoning in differential geometry and mechanics: part II, mechanical theorem proving. J. of Automated Reasoning, 1993, 10: 173–189

19 Chou S C., Gao X S. Automated reasoning in differential geometry and mechanics: part III, mechanical formula derivation. IFIP Transaction on Automated Reasoning, 1992, 1–12

20 Chou S C., Gao X S. Automated reasoning in differential geometry and mechanics: part IV, Bertrand curves. Sys. Sci and Math. Sci, 1993, 6: 186–192

21 Chou S C., Gao X S. Mechanical formula derivation in elementary geometries. In: Proc. IS-

SAC'90. New York: ACM, 1990,265–270

22 Chou S C., Gao X S. Proving constructive geometry statements. In: D. Kapur (eds.). Proc. CADE-11. LNCS 607. Springer-Verlag, 1992, 20–34

23 Chou S C., Gao X S. Ritt-Wu's decomposition algorithm and geometry theorem proving. In: M.E. Stickel (eds.). Proc. CADE-10. LNCS 449. Springer-Verlag, 1990, 207–220

24 Chou S C., Gao X S., Zhang J Z. Machine Proofs in Geometry. Singapore: World Scientific, 1994

25 Chou S C., Gao X S., Arnon D S. On the mechanical proof of geometry theorems involving inequalities. Advances in Computing Research, 1992, 6: 139–181

26 Davenport J H. On the integration of algebraic functions. In: LNCS 102. New York: Springer-Verlag, 1981

27 Fan E G. Extended tanh-function method and its applications to nonlinear equations. Physics Letters A, 2000, 277: 212–218

28 范恩贵. 数学机械化丛书: 可积系统与计算机代数. 科学出版社, 2004

29 Feng R., Gao X S. Polynomial general solutions of first order autonomous ODEs. In: LNCS 3519. Berlin: Springer, 2005, 7–19

30 Feng R., Gao X S. Rational general solutions of algebraic ordinary differential equations. In: Proc. ISSAC'04. New York: ACM Press, 2004, 155–162

31 Fine H. On the functions defined by differential equations, with an extension of the Puiseux Polygon construction to these equations. Amer. Jour. of Math., 1889, XI: 317–328

32 Gao X S. Implicitization for differential rational parametric equations. J. of Symb. Comput, 2003, 36(5): 811–824

33 Gao X S., Chou S C. Computations with parametric equations.In: Proc. ISSAC'91. New York: ACM Press. 122–127

34 Gao X S., Chou S C. Implicitization of rational parametric equations. J. of Symb. Comput, 1992, 14: 459–470

35 Gao X S., Chou S C. On the dimension for arbitrary ascendingChains. Chinese Bull. of Scis, 1993, vol. 38: 396–399

36 Gao X S., Chou S C. On the parameterization of algebraic curves. AAECC, 1992, 3: 27–38

37 Gao X S., Chou S C. Solving parametric algebraic systems. In: Proc. ISSAC'92. ACM Press, 1992, 335–341

38 Gao X S., Wang D K. Zero decomposition tree for counting the number of solutions for algebraic parametric equation Systems. In: Z. Li and W. Sit (eds.). Computer Mathematics III. Singapore: World Scientific, 2003, 130–145

39 Gao X S., Zhang G. Geometric constraint solving via C-tree decomposition. In: Proc. ACM SM03. Seattle, USA. New York: ACM Press, 2003, 45–55

40 Gao X S., Chou S C. A zero structure theorem for differential parametric systems. J. of Symb. Comput., 1994, 16: 585–595

41 Gao X S., Chou S C. On the normal parameterization of curvesand surfaces. IJCGA, 1991, 1: 125–136

42 Gao X S., Chou S C. On the theory of resolvents and its applications. Sys. Sci. and Math. Sci., 1999, 12, Suppl.: 17–30

43 Gao X S., Lei D., Liao Q., Zhang G. Generalized Stewart-Gough platforms and their direct kinematics. IEEE Trans. Robotics, 2005, 21(2): 141–151

44 Ge J., Chou S C., Gao X S. Geometric constraint satisfaction using optimization methods. Computer Aided Design, 2000, 31(14): 867–879

45 Gonzalez-Vega L., Rouillier F., Roy M F., Trujillo G. Symbolic recipes for real solutions. In:

Some Tapas of Computer Algebra. Berlin Heidelberg: Springer, 1999
46 Grigoriev D Y., Singer M. Solving ordinary differential equations in terms of series with real exponents. Trans. AMS, 1991, 327: 329–351
47 谷超豪, 胡和生, 周子翔. 孤立子理论中的达布变换及其几何应用. 上海科学技术出版社, 1999
48 Hilbert D. The Foundations of Geometry. Lasalla, Illinois: Open Court Pub. Comp., 1971
49 Hirota R. Exact solution of the Korteweg-deVries equation for multiple collisions of solitons. Phys. Rev. Lett., 1971, 27: 1192–1194
50 Holland J H. Adaptation in Nature and Artificial Systems. Ann Arbor: University of Michigan Press, 1975
51 Hubert E. Factorization-free decomposition algorithms in differential algebra. J. of Symb. Comput., 2000, 29: 641–662
52 Hubert E., Le Roux N. Computing power series solutions of a nonlinear PDE system. In: Proc. ISSAC'03. New York: ACM Press, 2003, 148–155
53 Hubert E., Le Roux N. Resolvent representation for regular differential ideals. AAECC, 2003, 13: 395–425
54 Kalkbrener M. A generalized euclidean algorithm for computing triangular representations of algebraic varieties. J. of Symb. Comput., 15: 143–167
55 Kapur D., Mundy J L. Wu's method and its application to perspective viewing. Artificial Intelligence, 1988, 37: 15–36
56 Kolchin E R. Differential Algebra and Algebraic Groups. New York: Academic Press, 1973
57 Kovacic J J. An algorithm for solving second order linear homogeneous differential equations. J. of Symb. Comput., 1986, 2(1): 3–43
58 Lan H B., Wang K L. Exact solutions for two nonlinear equations. I. J. Phys. A, 1990, 23: 3923–3928
59 Lazard D. A new method for solving algebraic systems of positive dimension. Discrete Appl. Math., 1991, 33: 147–160
60 Lazard D. Solving zero-dimensional algebraic systems. J. of Symb. Comput., 1992, 13: 117–131
61 Li H., Wu Y. Automated theorem proving in projective geometry with Cayley and bracket algebras, I. incidence geometry. J. of Symb. Comput., 2004, 36: 717–762
62 Li H., Wu Y. Automated theorem proving in projective geometry with Cayley and bracket algebras, II. conic geometry. J. of Symb. Comput., 2004, 36: 763–809
63 Li Z., Schwarz F., Tsarev S. Factoring linear partial differential systems with finite-dimensional solution spaces. J. of Symb. Comput., 2003, 36: 443–471
64 Li Z., Schwarz F. Rational solutions of Ricatti-like partial differential equations. J. of Symb. Comput., 2001, 31: 691–716
65 陆征一, 何璧, 罗勇. 多项式系统的实根分离算法及其应用. 科学出版社, 2004
66 Mishra B. Algorithmic Algebra. Berlin: Springer, 1993, 297–381
67 Moré J J., Wright S J. Optimization Software Guide. SIAM Press, 1993
68 Nemhauser G L., Rinnooy Kan A H G. Todd M J. (eds.), Optimization. Elsevier Science Publishers B.V., 1989
69 Olver P J. Application of Lie Groups to Differential Equations. New York: Springer-Verlag, 1999
70 Prelle M J., Singer M F. Elementary first integrals of differential equations. Trans. AMS, 1983, 279(1): 215–229
71 Risch R H. The problem of integration in finite terms. Trans. AMS, 1969, 139: 167–189
72 Ritt J F. Differential Algebra. Amer. Math. Sco. Colloquium, 1950
73 Rogers C., Shadwick W R. Bäcklund Transformation and Their Applications. New York: Academic Press, 1982

74 Singer M F. Liouillian solutions of nth order homogeneous linear differential equations. Amer. J. Math., 1981, 103(4): 661–682

75 Singer M F., Ulmer F. Liouvillian and algebraic solutions of second and third order linear differential equations. J. of Symb. Comput., 1993, 16: 37–73

76 Wang D K., Zhang Y. An algorithm for decomposing polynomial system into normal ascending sets. MM Preprints, 2004, 23

77 Wang D K. Polynomial equations solving and mechanical geometric theorem proving. PhD. Thesis. KLMM. Academia Sinica, 1993

78 Wang D M. Elimination Theory. Wien: Springer, 2002

79 Wang D M. Computing triangular systems and regular systems. J. of Symb. Comput., 2000, 30: 221–236

80 Wang D M. Decomposing polynomial systems into simple systems. J. of Symb. Comput., 1998, 25: 295–314

81 Wang J M., Gao X S. An algorithm for solving partial differential parametric systems. Discrete Applied Mathematics, 2004, 136(1): 105–116

82 Weiss J., Tabor M., Carnvale G. The Painleve property for partial differential equations. J. Math. Phys., 1983, 24: 522

83 Wu W T., Wang D K. On the Surface Fitting Problems in CAGD. Mathematics in Practice and Theory, 1994, 3 (in Chinese)

84 Wu W T. On surface-fitting problem in CAGD. MM Preprints, 1993, 10: 1–10

85 Wu W T. On the decision problem and the mechanization of theorem in elementary geometry. Scientia Sinica, 1978, 21: 159–172. Also in Automated Theorem Proving: After 25 Years. Contemporary Mathematics, AMS, 1984, 29: 213–234

86 Wu W T. Basic Principles of Mechanical Theorem Proving in Geometries, Volume I: Part of Elementary Geometries. Beijing: Science Press, 1984(in Chinese); Berlin: Springer, 1995(English Version)

87 Wu W T. Mathematics Mechanization. Science Press/Kluwer, 2000

88 Wu W T. A Constructive Theory of Differential Algebraic Geometry. In: Proc. of DD6-Symposium. Shanghai, 1984, 497–528

89 Wu W T. A mechanization method of geometry and its applications III. mechanical proving of polynomial inequalities and equation-solving. Sys. Sci. and Math. Sci., 1988: 1–17

90 Wu W T. A mechanization method of equations-solving and theorem-proving. Advances in Computing Research , 1992, 6: 103–138. Greenwich USA: JAI Press Inc.

91 Wu W T. A mechanization method of geometry and its applications II. curve pairs of Bertrand type. Kexue Tongbao, 1986, 17: 1281–1284

92 Wu W T. A mechanization method of geometry and its applications I. distances, areas and volumes. Sys. Sci. and Math. Scis., 1986, 6: 204–216

93 Wu W T. Basic principles of mechanical theorem-proving in elementary geometries. Sys. Sci. and Math. Sci., 1984, 4: 207–235. Re-published in J. Automated Reasoning, 1986, 2: 221–252

94 Wu W T. Mechanical theorem proving in elementary differential geometry. Scientia Sinica, 1979, 94–102(in Chinese)

95 Wu W T. Mechanical theorem proving of differential geometries and some of its applications in mechanics. J. Automated Reasoning, 1991, 7: 171–191

96 Wu W T. On a finiteness theorem about optimization problems. MM Preprints, 1992, 8: 1–18

97 Wu W T. On a projection theorem of quasi-varieties in elimination theory. Chinese Ann. of Math., 1990, 11B: 220–226

98 Wu W T. On reducibility problem in mechanical theorem proving of elementary geometries.

Chinese Quarterly J. of Math., 1987, 2: 1–19
99 Wu W T. On the chemical equilibrium problem and equation-solving. Acta Math. Sci., 1990, 10: 361–374
100 Wu W T. On the foundation of algebraic differential geometry. Sys. Sci. and Math. Scis., 1989, 2: 289–312
101 Wu W T. On zeros of algebraic equations-an application of Ritt principle. Kexue Tongbao, 1986, 31: 1–5
102 Wu W T. Toward mechanization of geometry-some comments on Hilbert's "Grundlagen der Geometrie". Acta Math. Scientia, 1982, 2: 125–138
103 Yan Z Y. New explicit travelling wave solutions for two new integrable coupled nonlinear evolution equations. Physics Letters A, 2001, 292: 100–106
104 Yang L., Zhang J Z. Searching dependency between algebraic equations: an algorithm applied to automated reasoning. ICTP preprint, IC/91/6. Trieste
105 杨路, 张景中, 侯晓荣. 非线性代数方程组与定理机器证明. 上海教育出版社, 1996
106 杨路, 侯晓荣, 夏壁灿. 自动发现不等式型定理的一个完备算法. 中国科学 E 辑, 2001, 3
107 袁春明. 续代数扩域上多项式因式分解的 Trager 算法. 系统科学与数学, 2006
108 赵新超. 进化计算研究及其应用. 中科院数学与系统科学研究院. 博士毕业论文, 2005
109 Zhi L., Reid G., Tang J. A complete symbolic-numeric linear method for camera pose determination. In: Proc. ISSAC2003. ACM Press, 2003, 215–223

附录　几何命题的描述

本章详细介绍 MMP 可以接受的几何命题的四种输入形式.

A.1　几何命题的谓词形式

几何命题的谓词形式如下：

```
Stat = [pvs,mvs,pts,ps,ds,CONC]
```

其中 pvs 是命题中的自由变量表, mvs 是命题中的非自由变量表, pts 是命题中的点的表列, ps 是代表命题中的等式形的几何关系或几何谓词, ds 是代表命题中的非等式形的几何关系, CONC 是命题的结论.

1. **几何量与代数表达式.** 点是我们使用的唯一的基本几何对象. 这里考虑的几何谓词都是关于点的函数. 平面上的一个点用一个英文字母开头后跟数字、字母或下画线的一个长度最多为九的字符串表示. 例如, A, B1, C_a1 均为合法的点的表示法.

面积　[area,A,B,C]和 [area,A,B,C,D] 分别用来表示三角形 ABC 和四边形 ABCD 的带号面积. 所以我们有

[area,A,B,C] = [area,B,C,A] = [area,C,A,B] = −[area,A,C,B]= −[area,C,B,A] =−[area,B,A,C].

[area,A,B,C,D] =− [area,C,B,A,D].

勾股差　三角形 ABP 和四边形 ABPQ 的勾股差分别定义为

[pdiff,A,B P] = $(AB)^2 + (BP)^2 - (PA)^2$.

[pdiff,A,B,P,Q] = $(AB)^2 - (BP)^2 + (PQ)^2 - (QA)^2$.

比例　设 A,B,P,Q 是四个点, 满足 A,B,P,Q 共线或 AB//PQ. 我们定义 [ratio, A, B, P, Q] 为有向线段 AB 与 PQ 的比值. 所以有

[ratio,A,B,P,Q] = − [ratio,B,A,P,Q] = [ratio,B,A,Q,P] = − [ratio,A,B,Q,P].

向量　平面上从原点到点 A 的向量用 [vec,A] 来表示. [vec,A,B] = [vec,B] − [vec,A].

全角　直线 AB 与直线 PQ 形成的全角用 [angle,A,B,P,Q] 表示.

距离平方　两点 A,B 的距离平方用 [sqdis,A,B] 表示.

交比　向量 AB 与 PQ 的交比用 [cratio,A,B,P,Q] 表示.

[cratio,A,B,P,Q] = [cratio,P,A,P,B]× [cratio,Q,B,Q,A].

变量　任何由字符、数字、符号和 {,}构成的字符串均作为变量. 例如, r,w,a1 等均为变量.

点的坐标 点的坐标可以由用户指定也可以由软件自动生成. 有两种类型的坐标. 在证明中, 一种坐标表示为 x1,x2,x3,$\cdots$, 另一种坐标表示为 u1,u2,u3,$\cdots$.

2. 几何谓词.

二维情形

设 $P_i = [x_i, y_i], i = 1, 2, \cdots$ 为欧氏平面上的点. 以下是我们要使用的几何关系. 对于每一个几何关系, 我们还将给出其等价的几何不变量形式或坐标形式. 也就是说每一个几何关系都能转化成相应的代数表达式.

[**coll{或 COLL},**P_1, P_2, P_3]: P_1, P_2, P_3 三点共线

几何不变量形式：$[area\ P_1, P_2, P_3] = 0$

坐标形式：$(x_1 - x_2) * (y_2 - y_3) - (x_2 - x_3) * (y_1 - y_2) = 0$

[**para{或 PARA},**P_1, P_2, P_3, P_4]：$(P_1 = P_2)$ 或 $(P_3 = P_4)$ 或 (P_1, P_2, P_3, P_4 共线) 或 $(P_1P_2 \parallel P_3P_4)$

几何不变量形式：[area P_1, P_3, P_2, P_4] = 0

坐标形式：$(x_1 - x_2) * (y_3 - y_4) - (x_3 - x_4) * (y_1 - y_2) = 0$

注：这里的两线平行的概念包含了许多的特殊情形. 当 $P_1 = P_2$ 或 $P_3 = P_4$ 及 P_1, P_2, P_3, P_4 共线时我们也称 P_1P_2, P_3P_4 平行. 这样做的目的是为了得到简洁的代数表达式.

[**perp{或 PERP},**P_1, P_2, P_3, P_4]：$(P_1 = P_2)$ 或 $(P_3 = P_4)$ 或 $(P_1P_2 \perp P_3P_4)$

几何不变量形式：[pdiff P_1, P_3, P_2, P_4] = 0

坐标形式：$(x_1 - x_2) * (x_3 - x_4) + (y_3 - y_4) * (y_1 - y_2) = 0$

[**cong{或 CONG},**P_1, P_2, P_3, P_4]：线段 P_1P_2 与 P_3P_4 全等

几何不变量形式：[sqdis,P_1, P_2]=[sqdis,P_3, P_4]

坐标形式：$(x_1 - x_2)^2 + (y_1 - y_2)^2 - (x_3 - x_4)^2 - (y_3 - y_4)^2 = 0$

[**acong{或 ACONG},**$P_1, P_2, P_3, P_4, P_5, P_6$]：全角 $P_1P_2P_3$ 与全角 $P_4P_5P_6$ 全等

坐标形式：[coll,P_2, P_1, P_3]*[perp,P_5, P_4, P_5, P_6] = [coll,P_5, P_4, P_6]*[perp,P_2, P_1, P_2, P_3]

[**cir{或 CIR},**P_1, P_2, P_3, P_4]：四点 P_1, P_2, P_3, P_4 共圆

等价形式：[acong,$P_1, P_3, P_2, P_1, P_4, P_2$]

[**tangent{或 TANGENT},**P_1, P_2, P_3, P_4]：(P 圆 P_1P_2) 与 (P 圆 P_3P_4) 相切

坐标形式：([sqdis,P_1, P_3] − [sqdis,P_1, P_2] − [sqdis,P_3, P_4])^2 = 4*[sqdis, P_1, P_2]*[sqdis, P_3, P_4]

[**midx{或 MIDX},** P_1, P_2, P_3]：点 P_1 是 P_2, P_3 在 x 轴上的中点

坐标形式：$2 * x_1 - x_2 - x_3 = 0$

[**midy{或 MIDY},** P_1, P_2, P_3]：点 P_1 是 P_2, P_3 在 y 轴上的中点

坐标形式：$2*y_1-y_2-y_3=0$

[**eq_prod{或 EQ_PROD},**$P_1,P_2,P_3,P_4,P_5,P_6,P_7,P_8$]：

线段长度关系为 $|P_1P_2|*|P_3P_4|=|P_1P_2|*|P_3P_4|$

坐标形式：[sqdis,P_1,P_2]* [sqdis,P_1,P_2] =[sqdis,P_5,P_6]*[sqdis,P_7,P_8]

[**area,var,**$P_1,P_2,\cdots,P_n$]：变元 var 是 $n-2$ 个三角形的有向面积和

坐标形式：var=([area, P_1,P_2,P_3] +[area, P_1,P_3,P_4] $+\cdots+$ [area, P_1,P_{n-1},P_n])

[**crat,var,**P_1,P_2,P_3,P_4]：变元 var 是 P_1P_2,P_3P_4 的交比

坐标形式：var$*(x_4-x_1)*(x_2-x_3)-(x_3-x_1)*(x_2-x_4)=0$

[**dis,**P_1,P_2]：线段 P_1P_2 的平方距离

坐标形式：$(x_1-x_2)^2+(y_1-y_2)^2=0$

[**dis,**P_1,P_2**,var**]：变元 var 是线段 P_1P_2 的距离. 如果没有 var, 则同 [dis,P_1,P_2]

坐标形式：var^2=[sqdis,P_1,P_2]

[**displ,**P_1,P_2,P_3**,var**]：变元 var 是点 P_1 到直线 P_2P_3 的距离

坐标形式：var^2*[sqdis,P_2,P_3]=[coll,P_2,P_3,P_1]^2

[**xrat, var,** P_1,P_2,P_3,P_4]：变元 var 是四点 x 轴坐标之差的比率

坐标形式：var$*(x_3-x_4)-(x_1-x_2)=0$

[**yrat, var,** P_1,P_2,P_3,P_4]：变元 var 是四点 y 轴坐标之差的比率

坐标形式：var$*(y_3-y_4)-(y_1-y_2)=0$

[**eqx,**P_1,P_2]：P_1,P_2 的 x 轴坐标相等

坐标形式：$x_1-x_2=0$

[**eqy,**P_1,P_2]：P_1,P_2 的 y 轴坐标相等

坐标形式：$y_1-y_2=0$

[**nminus,**$pol_1,pol_2,\cdots,pol_n$]：$pol_1,pol_2,\cdots,pol_n$ 为多项式, 结果为 $pol_1-(pol_2+\cdots+pol_n)$

[**nsum,**$pol_1,pol_2,\cdots,pol_n$]：$pol_1,pol_2,\cdots,pol_n$ 同上, 结果为 $pol_1+pol_2+\cdots+pol_n$

[**nprod,**$pol_1,pol_2,\cdots,pol_n$]：$pol_1,pol_2,\cdots,pol_n$ 同上, 结果为 $pol_1*pol_2*\cdots*pol_n$

[**npower,pol,n**]：pol 为多项式, n 为正整数, 结果为 pol^n

三维情形

设 $V_i=[x_i,y_i,z_i],i=1,2,\cdots,$ 为欧氏空间上的点、向量或曲线. 以下是我们要使用的几何关系. 对于每一个几何关系, 我们还将给出其等价的几何不变量形式或坐标形式. 也就是说每一个几何关系都能转化成相应的代数表达式.

3. 关于向量的谓词.

[**exprod,**V_1,V_2]：向量 V_1,V_2 的外积, $V_1\times V_2$

坐标形式：$[y_1*z_2-z_1*y_2,x_2*z_1-x_1*z_2,x_1*y_2-y_1*x_2]$

[**inprod,**V_1, V_2]：向量 V_1, V_2 的内积, $V_1 \cdot V_2$

坐标形式：$x_1 * x_2 + y_1 * y_2 + z_1 * z_2$

[**comprod,**V_1, V_2, V_3]：向量 V_1, V_2 和 V_3 的混合积,$V_1 \cdot (V_2 \times V_3)$

坐标形式：$x_1 * (y_2 * z_3 - z_2 * y_3) + y_1 * (x_3 * z_2 - x_2 * z_3) + z_1 * (x_2 * y_3 - y_2 * x_3)$

[**numprod,**V_1, p]：向量 V_1 和多项式 p 的数乘, $V_1 \cdot p$

坐标形式：$[x_1 * p, y_1 * p, z_1 * p]$

[**vecplus,**$V_1, V_2, \cdots, V_n$]：$V_1 + V_2 + \cdots + V_n$

坐标形式：$[x_1+, \cdots, +x_n, y_1+, \cdots, +y_n, z_1+, \cdots, +z_n\ ]$

[**vecminus,**$V_1, V_2, \cdots, V_n$]：$V_1 - V_2 - \cdots - V_n$

坐标形式：$[x_1-, \cdots, -x_n, y_1-, \cdots, -y_n, z_1-, \cdots, -z_n\ ]$

[**vecdiff,**V_1**,n{defaut=1},var{defaut=0}**]：向量 V_1 关于变元 var 的第 n 阶导数

坐标形式：[Diff(x_1,n,var),Diff(y_1,n,var),Diff(z_1,n,var)]

[**vecneg,**V_1]：向量 V_1 的相反方向向量

坐标形式：[−x1, −y1, −z1]

向量的几何关系

[**v_para,**V_1, V_2]：向量 V_1 和 V_2 平行

坐标形式：[exprod, V_1, V_2] = 0

[**v_perp,**V_1, V_2]：向量 V_1 和 V_2 垂直

坐标形式：[inprod, V_1, V_2]=0

[**cons_v,**$V_1, V_2, \cdots, V_n$]：向量 $V_1, V_2, \cdots, V_n$ 都是常向量

坐标形式：Diff(V_1)=,$\cdots$,=Diff(V_n)=0

[**cons_var,**$var_2, \cdots, var_n$]：变元或多项式 $var_2, \cdots, var_n$ 都是常向量

[**cons_dir,**V_1]：向量 V_1 方向固定

坐标形式：wronskian(x_1, y_1, z_1)=0

[**v_norm,**V_1**,var**]：变元 var 等于向量 V_1 范数的平方

坐标形式：var$-x_1^2 - y_1^2 - z_1^2$

[**cons_len,**V_1]：向量 V_1 定长

坐标形式：$x_1 * x_1' + x_1 * x_1' + x_1 * x_1'$

[**perp_fix_line,**V_1]：向量 V_1 垂直于给定的一条直线

坐标形式：wronskian(x_1, y_1, z_1)=0

[**para_fix_plane,**V_1]：向量 V_1 平行于给定的一个平面

坐标形式：wronskian(x_1, y_1, z_1)=0

[**co2_linear,**V_1, V_2, V_3]：点 V_3 在经过点 V_2 平行于向量 V_1 的直线上

坐标形式：$V_1 * (V_3 - V_2)$=0

[**fix_co2_linear,**V_1, V_2]：所有经过点 V_2 平行于向量 V_1 的直线过一定点

坐标形式：wronskian$(x_1, y_1, x_1 * y_2 - y_1 * x_2)$

=wronskian$(x_1, z_1, x_1 * z_2 - y_1 * x_2)$

=wronskian$(y_1, z_1, x_1 * z_2 - y_1 * y_2)$=0

[**co3_linear,**V_1, V_2, V_3]：点 V_1, V_2 和 V_3 共线

坐标形式：$(V_2 - V_1) * (V_3 - V_1)$=0

[**fix_co3_linear,**V_1, V_2]：所有经过点 V_1 和 V_2 的直线过一定点

同 [fix_co2_linear,$V_2 - V_1, V_1$]

[**co2_plane,**V_1, V_2, V_3]：向量 V_3 在经过向量 V_2 以 V_1 为法向量的平面上

坐标形式：$V_1 \cdot (V_3 - V_1)$=0

[**fix_co2_plane,**V_1, V_2]：所有经过向量 V_2 且以 V_1 为法向量的平面过一定点

坐标形式：wronskian$(x_1, y_1, z_1, x_1 * x_2 + y_1 * y_2 + z_1 * z_2)$=0

并且 [perp_fix_line,V_1]!=0

[**co3_plane,**V_1, V_2, V_3]：向量 V_1, V_2 和 V_3 平行于某个平面

坐标形式：$V_1 \cdot (V_2 * V_3)$=0

[**fix_co3_plane,**V_1, V_2]：所有包含原点和点 V_1, V_2 的平面过一定点

坐标形式：wronskian$(y_1 * z_2 - z_1 * y_2, -x_1 * z_2 + z_1 * x_2, x_1 * y_2 - y_1 * x_2)$=0

[**co4_plane,**V_1, V_2, V_3, V_4]：点 V_1, V_2, V_3 和 V_4 在同一平面上

同 [co3_plane,$V_2 - V_1, V_3 - V_1, V_4 - V_1$]

[**fix_co4_plane,**V_1, V_2, V_3]：点 V_1, V_2 和 V_3 决定的平面过一定点

坐标形式：[fix_co2_plane,$V_1 * V_2, V_3$], 并且 [perp_fix_line,$V_1 * V_2$]!=0

[**angle,**V_1, V_2**,var**]：变元 var 等于向量 V_1 和 V_2 的内积

坐标形式：var$- x_1 * x_2 + y_1 * y_2 + z_1 * z_2$

[**fix_angle,**V_1]：向量 V_1 和给定的一个方向的夹角是个常数

坐标形式：wronskian$((y_1^2 + z_1^2) * x_1' - x_1 * (y_1 * y_1' + z_1 * z_1'),$

$(x_1^2 + z_1^2) * y_1' - y_1 * (x_1 * x_1' + z_1 * z_1'),$

$(x_1^2 + y_1^2) * z_1' - z_1 * (x_1 * x_1' + y_1 * y_1'))$=0

[**diff,v{或 poly},n{defaut=1},var{defaut=0}**]：变元 v 或多项式 poly 关于变元 var 的第 n 阶导数

空间曲线的几何关系

[**curve,**V_1, var_1, var_2, var_3]：变元 var_2 和 var_3 分别是曲线 V_1 的曲率和挠率, var_1 是曲线 V_1 的切向量的平方

坐标形式：$var_1 - {z_1'}^2 - {y_1'}^2 - {x_1'}^2 = 0$

$var_1^3 * var_2^2 - (V_1' * V_1'').(V_1' * V_1'') = 0$

$var_1^3 * var_2^2 * var_3 - V_1'.(V_1'' * V_1''') = 0$

[**curve_arc,**V_1, var_1, var_2]：变元 var_2 和 var_3 分别是以弧长为参数的曲线 V_1 的曲率和挠率

坐标形式：$var_1^2 - x_1''2 - y_1''2 - z_1''2 = 0$

$var_1^2 * var_2 - V_1' \cdot (V_1'' * V_1''') = 0$

[**curve_norm,**V_1]：曲线 V_1 的主法向量

坐标形式：$(V_1' \cdot V_1') * V_1'' - (V_1' \cdot V_1'') * V_1'$

[**curve_binorm,**V_1]：曲线 V_1 的次法向量

坐标形式：V_1'*[curve_norm,V_1]

[**fix_line,**V_1]：曲线 V_1 是一条直线

坐标形式：同 [cons_dir,V_1']

[**fix_plane{或 FIX_PLANE},**V_1]：曲线 V_1 在一个平面内

坐标形式：$V_1' \cdot (V_1'' * V_1''') = 0$

或 wronskian(1,x_1, y_1, z_1)=0

[**fix_plane_o或 FIX_PLANE_O,**V_1]：曲线 V_1 在过原点的一个平面内

坐标形式：wronskian(x_1, y_1, z_1)

[**fix_sph或 FIX_SPHERE,**V_1]：曲线 V_1 在一个球面上

坐标形式：wronskian($x_1', y_1', z_1', x_1 * x_1' + y_1 * y_1' + z_1 * z_1'$)=0

并且 wronskian(x_1', y_1', z_1')$\neq 0$

[**fix_helix或 FIX_HELIX,**V_1]：曲线 V_1 是螺旋型曲线

坐标形式：$V_1'' \cdot (V_1''' * V_1'''')$=0

[**xy_circle或 XY_CIRCLE,**V_1]：曲线 V_1 在 xy 平面上的投影是一个圆

坐标形式：wronskian(1,x,y,$x^2 + y^2$)=0

A.2 几何命题的构造形式

构造型几何命题是几何命题的一种很自然的描述. 其特点是用关于直线与圆的作图语句来描述一幅几何图形.

1. 点, 线, 圆.

点 点是我们使用的唯一的基本几何对象. 其他的几何对象, 如直线、圆、三角形等均表示为点的函数. 平面上的一个点用一个英文字母开头后跟数字、字母或下画线的一个长度最多为九的字符串表示. 例如, A, B1, C_a1 均为合法的点的表示法.

直线 下面列出作图语句中所使用的直线：

[line,U,V] 过点 U 和 V 的直线.

[pline,W,U,V] 过点 W 且与直线 UV 平行的直线.

[tline,W,U,V] 过点 W 且与直线 UV 垂直的直线.

[aline,U,V,P,Q,R] 是一条通过点 U 的直线. 该直线与 UV 构成的全角与全角 ∠ [PQ,QR] 相等.

圆　我们使用两种圆：

[cir,O,U] 一个以 O 为圆心且通过点 U 的圆.

[ccir,O,U,V] 以 O 为圆心以 UV 为半径的圆.

2. 作图语句.　一个作图语句可以从一些已知的点作出一个新的点. 例如, 已知两个点则可以作它们的中点. 对每一个作图语句, 我们将给出其几何关系形式的定义, 同时给出其非退化条件. 非退化条件是为了保证作图语句能作出真正的点. 例如, 当我们作两线交点时, 非退化条件就是这两线不能平行.

以下分类列出作图语句：作自由点的语句, 作半自由点的语句, 作交点的语句, 作比值的语句, 作三角形的特殊点的语句, 作常数的语句.

自由点　以下列出作自由点的作图语句：

point,A,B,...,H：作出自由点 A, B, $\cdots$, H.

circle,O,A,B,...,H：作出一个圆心为 O 的圆上的点 A, B, $\cdots$, H.

半自由点　以下列出作半自由点的作图语句：

[on,P,O]：在几何体 O 上取点 P. 具体有如下形式：

1° 在 L 线上取点 [on,P,[line,A,B]]：在直线 AB 上取点 P.

几何关系形式：[coll,P,A,B].

非退化条件：A $\neq$ B.

2° 在 P 线上取点 [on,P,[pline,A,B,C]]：在 [pline,A,B,C] 上取一点 P.

几何关系形式：[para,P,A,B,C].

非退化条件：B$\neq$C.

3° 在 T 线上取点 [on,P,[pline,A,B,C]]：在 [tline,A,B,C] 上取一点 P.

几何关系形式：[perp,P,A,B,C].

非退化条件：B$\neq$ C.

4° 在 A 线上取点 [on,Y,[aline,A,B,P,Q,U]]：在 [aline,A,B,P,Q,U] 上取一点 Y.

几何关系形式：[acong,Y,A,B,P,Q,U].

非退化条件：$A \neq B, P \neq Q, Q \neq U$.

5° 在 P 圆上取点 [on,P,[cir,A,B]]：在圆 [cir,A,B] 上取一点 P.

几何关系形式：[cong,P,A,B,A].

非退化条件：$A \neq B$.

6° 在 R 圆上取点 [on,P,[ccir,A,B,C]]：在圆 [ccir,A,B,C] 上取一点 P.

几何关系形式：[cong,A,P,B,C].

非退化条件：$B \neq C$.

交点　以下列出作交点的作图语句：

1° L 线与 L 线交点 [inter,Y,[line,A,B],[line,P,Q]]：

作直线 AB 与 PQ 的交点,

即 Y= [line,A, B] ∩ [line,P,Q].

几何关系形式：[coll,Y,A,B] 且 [coll,Y,P,Q].

非退化条件：AB 不平行于 PQ.

2° L 线与 P 线交点 [inter,Y,[line,A,B],[pline,P,Q,R]]：

作 [line,A,B] 与 [pline,P,Q,R] 的交点,

即 Y = [line,A,B] ∩ [pline,P,Q,R].

几何关系形式：[coll,Y,A,B] 且 [PARA,Y,P,Q,R].

非退化条件：AB 不平行于 QR.

3° L 线与 T 线交点 [inter,Y,[line,A,B],[tline,P,Q,R]]：

作 [line,A,B] 与 [tline,P,Q,R] 的交点,

即 Y = [line,A,B] ∩ [tline,P,Q,R].

几何关系形式：[coll,Y,A,B] 且 [coll,Y,P,Q,R].

非退化条件：AB 不垂直于 QR.

4° P 线与 P 线交点 [inter,Y,[pline,A,B,C],[pline,Q,U,V]]：

作 [pline,A,B,C] 与 [pline,Q, U,V] 的交点,

即 Y = [pline,A,B,C] ∩ [pline,Q,U,V].

几何关系形式：[PARA,Y,A,B,C] 且 [PARA,Y,Q,U,V].

非退化条件：BC 不平行于 UV.

5° P 线与 T 线交点 [inter,Y,[pline,A,B,C],[tline,Q,U,V]]：

作 [pline,A,B,C] 与 [tline,[Q, U,V] 的交点,

即 Y = [pline,A,B,C] ∩ [tline,Q,U,V].

几何关系形式：[PARA,Y,A,B,C] 且 [perp,Y,Q,U,V].

非退化条件：BC 不垂直于 UV.

6° T 线与 T 线交点 [inter,Y,[tline,A,B,C],[tline,Q,U,V]]：

作 [tline,A,B,C] 与 [tline, U,V] 的交点,

即 Y = [tline,A,B,C] ∩ [tline,Q,U,V].

几何关系形式：[perp,Y,A,B,C] 且 [perp,Y,Q,U,V].

非退化条件：BC 不平行于 UV.

7° L 线与 P 圆交点 [inter,Y,[line,A,B],[cir,P,Q]]：

作 [line,A,B] 与 [cir,P,Q] 的交点,

即 Y = [line,A,B] ∩ [cir,P,Q].

几何关系形式：[coll,Y,A,B] 且 [cong,Y,P,Q,P].

非退化条件：A≠ B 且 P ≠Q.

8° P 线与 P 圆交点 [inter,Y,[pline,A,B,P],[cir,Q,U]]：

作 [pline,A,B,P] 与 [cir,Q,U] 的交点,

即 Y = [pline,A,B,P] ∩ [cir,Q,U].

几何关系形式：[PARA,Y,A,B,P] 且 [cong,Y,Q,U,Q].

非退化条件：B≠ P 且 Q ≠ U.

9° T 线与 P 圆交点 [inter,Y,[tline,A,B,P],[cir,Q,U]]：

作 [tline,A,B,P] 与 [cir,Q,U] 的交点,

即 Y = [tline,A,B,P] ∩ [cir,Q,U].

几何关系形式：[perp,Y,A,B,P] 且 [cong,Y,Q,U,Q].

非退化条件：B≠ P 且 Q ≠ U.

10° P 圆与 P 圆交点 [inter,Y,[cir,A,B],[cir,P,Q]]：

作 [cir,A,B] 与 [cir,P,Q] 的交点,

即 Y = [cir,A,B] ∩ [cir,P,Q],

几何关系形式：[cong,Y,A,B,A] 且 [cong,Y,P,Q,P].

非退化条件：A≠ B 且 P ≠ Q.

11° L 线与 R 圆交点 [inter,Y,[line,A,B],[ccir,P,Q,R]]：

作 [line,A,B] 与 [ccir,P,Q,R] 的交点,

即 Y = [line,A,B] ∩ [ccir,P,Q,R].

几何关系形式：[coll,Y,A,B] 且 [cong,Y,P,Q,R].

非退化条件：A≠B, Q≠R.

12° P 圆与 R 圆交点 [inter,Y,[cir,A,B],[ccir,P,Q,R]]：

作 [cir,A,B] 与 [ccir,P,Q,R] 的交点,

即 Y = [cir,A,B] ∩ [ccir,P,Q,R].

几何关系形式：YA=AB, YP=QR.

非退化条件：A≠B, Q≠R.

13° R 圆与 R 圆交点 [inter,Y,[ccir,A,B,C],[ccir,P,Q,R]]：

作 [ccir,A,B,C] 与 [ccir,P,Q,R] 的交点,

即 Y = [ccir,A,B,C] ∩ [ccir,P,Q,R].

几何关系形式：YA=BC, YP=QR.

非退化条件：B≠C, Q≠R.

比例构造语句 以下列出有关比值的作图语句：

1° 中点 [midpoint,P,A,B]：

作出线段 AB 的中点 P.

几何关系形式：[midpoint,P,A,B].

2° 对称点 [sym,P,A,B]：

在直线 AB 上作出点 B 关于点 A 的对称点 P.

几何关系形式：[midpoint,A,P,B].

3° L 比例 [lratio,P,A,B,(e_1),(e_2)]：作出直线 AB 上一点 P, 它使得 [ratio,A,P,A,B] = e_1/e_2. 其中 e_1 和 e_2 均为代数表达式.

几何关系形式：[coll,P,A,B] 且 [ratio,A,P,A,B] = e_1/e_2.

非退化条件：A≠ B.

注：如果 e_2=1, 则可以省略 e_2. 这时本作图语句变成 "lratio P P1 P2 r", 它与作图语句 "lratio P P1 P2 (r)(1)" 相同. 此外, 如果 e_1 和 e_2 都有, 则我们需要使用括号把它们分别括起来. 例如, "lratio P P1 P2 r r" 与 "lratio P P1 P2 (r ∧ 2) (1)" 是相同的, 但与 "lratio P P1P2 (r) (r)" 不相同.

4° M 比例 [mratio,P,A,B,(e_1),(e_2)]：作出直线 AB 上一点 P, 它使得 [ratio,A,P,P,B] = e_1/e_2. 其中 e_1 和 e_2 均为代数表达式.

几何关系形式：[coll,P,A,B] 且 [ratio,A,P,P,B] = e_1/e_2.

非退化条件：A≠ B.

5° P 比例 [pratio,Y,A,B,C,(e_1),(e_2)]：作出直线 [pline,A,B,C] 上一点 Y, 它使得 [ratio,A,Y,B,C] = e_1/e_2. 其中 e_1 和 e_2 均为代数表达式.

几何关系形式：[para,Y,A,B,C] 且 [ratio,A,Y,B,C] = e_1/e_2.

非退化条件：B≠ P.

6° T 比例 [tratio,Y,A,B,C,(e_1),(e_2)]：作出直线 [tline,A,B,C] 上一点, 它使得 4[area,Y,B,A,C]/[pdiff,P,B,C,C,B]=YA/BC=e_1/e_2. 其中 e_1 和 e_2 均为代数表达式.

几何关系形式：[perp,Y,A,B,C] 且 4[area,Y,B,A,C]/[pdiff,B,C,C,B]
=YA/BC=e_1/e_2.

非退化条件：B≠ C.

7° 反演点 [inversion,Y,A,B,C]：

作出点 A 关于圆 [cir,B,C] 的反演点 Y.

几何关系形式：[inversion,Y,A,B,C].

非退化条件：B≠ C.

8° 调和点 [harmonic,Y,A,B,C]：

作出一点 Y 使得 (YA,BC) 形成一个调和对.

几何关系形式：[harmonic,Y,A,B,C].

非退化条件：A≠ B 且 B≠ C.

9° 正向正方形 [psquare,P,A,B]：作出一点 P, 使得 PA = AB, PA ⊥ AB, 并且从 PA 到 BA 的方向是反时针方向.

几何关系形式：[perp,Y,A,A,B] 且 4[area,P,A,B]=[pdiff,A,B,B,A].

非退化条件：A≠B.

10° 反向正方形 [nsquare,P,A,B]：作出一点 P, 使得 PA = AB, PA ⊥ AB, 并且从 PA 到 BA 的方向是顺时针方向.

几何关系形式：[perp,P,A,A,B] 且 4[area,P,A,B]=−[pdiff,A,B,B,A].

非退化条件：A≠ B.

11° 正三角形 [petriang,P,A,B]：作出一点 P, 使得 PAB 为一个等边三角形, 且从 PA 到 BA 的方向是反时针方向.

几何关系形式：[cong,P,A,P,B] 且 8[area,P,A,B]= [pdiff,A,B,B,A].

非退化条件：A≠ B.

三角形的特殊点　以下列出作三角形特殊点的作图语句：

1. 垂足 [foot,P,A,B,C]：P 是从点 A 向直线 BC 作的垂足. 本作图语句与 [inter, P,[line,B,C] ,[tline,A,B,C]] 等效.
2. 重心 [centroid,P,A,B,C]：作三角形 ABC 的重心 P.

   几何关系形式：3 * V(P) = V(A) + V(B) + V(C).
3. 垂心 [ocenter,P,A,B,C]：作三角形 ABC 的垂心 P. 本作图语句与 [inter,P,[tline, B,C, A],[tline,A,B,C]] 等效.
4. 反内心 [incenter,P,A,B,C]：作点 P 使得 C 是三角形 ABP 的内心 (或旁心).

   几何关系形式：[ccong,P,A,C,C,A,B] 且 [ccong,P,B,C,C,B,A].

   非退化条件：AC 不垂直 CB.

3. 构造型几何命题.

构造序列　一个构造序列是一列由已知的点作出新点的作图语句的序列. 任何一个构造序列都必须以作图命令 “POINT” 或 “CIRCLE” 开始. 例如,

```
[point,A,B,C];[inter,D,[pline,C,A,B],[pline,A,B,C]];
[inter,O,[line,A,C],[line,B,D]]
```

是一个作图序列. 而

```
[point,A,B]; [inter,D,[pline,C,A,B],[pline,A,B,C]];
```

`[inter,O,[line,A,C],[line,B,D]]`

不是一个作图序列. 这是因为第二个作图语句中的点 C 没有在前面某一个语句作出.

构造型几何命题 一个构造型几何命题具有以下形式:

`EXAMPLE NAME HYPOTHESES:c1;c2; ... cn; SHOW: c`

其中, {c1,c2,···,cn}是一个构造序列, c 是命题的结论. c 可以是下面三种形式之一:

- e1=e2, 其中 e1,e2 是两个代数表达式.
- 以下形式的一个几何关系:
  - ◇ [coll,A,B,C].
  - ◇ [PARA,A,B,P,Q].
  - ◇ [perp,A,B,P,Q].
  - ◇ [cong,A,B,P,Q].
  - ◇ [ccong,A,B,C,W,U,V].
  - ◇ [cocircle,A,B,C,D].
  - ◇ [midpoint,P,A,B]. P 是线段 AB 的中点.
  - ◇ [harmonic,A,B,P,Q]. (AB, PQ) 构成一个调和对.
  - ◇ [inversion,A,B,P,Q]. A 点是 B 点关于圆 [cir,P,Q] 的反演点.
  - ◇ [petriang,A,B,P]. PAB 是一个等边三角形.
  - ◇ [线段等积 A B P Q U V X]Y. 即 AB * PQ = UV * XY.
  - ◇ [tangent,A,B,P,Q]. 圆 [cir,A,B] 与圆 [cir,P,Q] 相切.
- 命题没有结论. 当用户的目的只是想画一个图, 而不想证明任何结论时, 可以将命题的结论置为 “[ ]”.

不可约构造型几何命题 一个不可约构造型几何命题是一个构造型几何命题, 它的每一个点都是唯一确定的. 在一个线性构造型几何命题中, 不能使用以下作图语句:

`L线与R圆交点,  P圆与R圆交点,  R圆与R圆交点.`

当使用以下作图语句:

`L线与P圆交点, P线与P圆交点, T线与P圆交点, P圆与P圆交点`

的时候, 只有下面的特殊形式被允许使用:

L 线与 P 圆交点 [inter,P,[line,A,B],[cir,C,A]]:

即 P = [line,A,B] ∩ [cir,C,A].

几何关系形式：[coll,P,A,B] 和 [cong,C,P,C,A].

非退化条件：C ≠ A, A≠ B 且 P ≠ A.

注：该作图语句的几何意义是, 点 A 既在直线上又在圆上. 作图语句作出直线 AB 与圆 [cir,P,A] 的不同于点 A 的那个点 P.

P 线与 P 圆交点　[inter,P,[pline,A,B,C],[cir,Q,A]]：

即 P = [pline,A,B,C] ∩ [cir, Q, A].

几何关系形式：[PARA,P,A,B,C] 且 [cong,P,Q,A,Q].

非退化条件：P≠ A, B≠ C 且 Q ≠ A.

T 线与 P 圆交点　[inter,P,[tline,A,B,C],[cir,Q,A]]：

即 P = [tline,A,B,C] ∩ [cir,Q,A].

几何关系形式：[perp,P,A,B,C] 且 [cong,P,Q,A,Q].

非退化条件：P≠ A, B≠ C 且 Q ≠ A.

P 圆与 P 圆交点　[inter,P,[cir,A,B],[cir,C,B]]：

即 P = [cir,A,B] ∩ [cir,C,B].

几何关系形式：[cong,P,A,B,A] 且 [cong,P,C,B,C].

非退化条件：P≠ B, A≠ B 且 C ≠ B.

4. 构造型几何命题的谓词形式. 几何命题的构造型描述对于作图与定理证明都很方便. 但却与我们通常所说的定理的“假设推得结论”的形式不完全吻合. 构造型几何命题的谓词形式即将这一命题转换为几何命题的谓词形式.

构造型几何命题的谓词形式由假设条件、非退化条件和结论构成.

假设条件 (HYP)　是一组由作图语句确定的几何关系.

非退化条件 (NDG)　是其中作图语句的非退化条件的集合, 以及命题结论表达式的分母不等于零所构成的条件.

命题结论　用 CONC 表示.

A.3　几何命题的自然语言形式

1. 自然语言中的几何对象.

点　一个点可以由以下语句产生：

```
point P1
the midpoint of P1P2
the foot from P1 to P2P3
the inversion of P1 with circle P2P3
the centroid of triangle P1P2P3
```

the circumcenter of triangle P1P2P3

the orthocenter of triangle P1P2P3

the incenter of triangle P1P2P3

线 一条线可以由以下格式给出：

line P1P2

the line passing through [point] P1 and parallel to[line] P2P3

the line passing through [point] P1 and perpendicular to [line] P2P3

the perpendicular bisector of [segment] P1P2

diagonal P1P2 (= line P1P2)

altitude P1P2 (= line P1P2)

median P1P2 (= line P1P2)

圆 一个圆可以由以下格式给出：

circle O

circle with center O

circle OP1

circle with center O and passing through point P1

the circumcircle [O] of [triangle] P1P2P3

the circumscribed circle [O] of [triangle] P1P2P3

the nine point circle [O] of [triangle] P1P2P3

the incircle [O] of [triangle] P1P2P3

the inscribed circle [O] of [triangle] P1P2P3

多边形 一个多边形可以由以下格式给出：

triangle P1P2P3

right triangle P1P2P3 (P1P2⊥P2P3)

isosceles triangle P1P2P3 (P1P2 = P2P3)

equilateral triangle P1P2P3

quadrilateral P1P2P3P4

parallelogram P1P2P3P4 (P1P2||P3P4, P1P3||P2P4)

trapezoid P1P2P3P4 (P1P2||P3P4)

rectangle P1P2P3P4

square P1P2P3P4

pentagon P1P2P3P4P5

组合形式 很多复杂的几何对象要用以下组合形式来给出：

num Point[s] on obj

1° num 可以是 one, two, three, four, five, six, seven, eight, nine, ten, eleven, twelve.

2° obj 可以是一条直线或一个圆.

这条语句说明了在 obj 上有 num 个点, 这句话的关键字是 on.

例子：

```
three points on the circumscribed circle O of triangle ABC.
a point on the line passing through point P1 and perpendicular
to median EF.
intersection obj1 and obj2
```

这条语句给出了对象 obj1 和 obj2 的交点,

这里 obj1 和 obj2 可以是一条直线或一个圆.

例子：

```
the intersection of line AB and the circle OA
(这里我们表示的交点不是A).
the intersection of circle OA and the circumcircle of triangle ABC.
```

2. 自然语言的格式. 一个完整的自然语言的几何语句有以下格式：

[Let] $exp_1, \cdots, exp_s$ verb obj.

1° exp_i 可以是一个点, 或是一串点. 所有的 exp_i 必须包含相同个数的点.

2° verb 可以是 be, is, are.

3° obj 是一个组合的对象.

Let 语句说明了一组新的几何对象 $exp_1, \cdots, exp_s$. 它们由 obj 来描述.

例子：

```
Let ABCD be a parallelogram.
A, B, C and D are four points on a circle.
P is the intersection of line AB and the circumcircle of triangle
P1P2P3.
```

自动证明不会检查语句是否符合英文语法. 单词 a, and, of, with, the, another, to 在证明过程中将被删掉. 在证明中以上三句分别和下面三句等价.

```
Let ABCD be parallelogram.
A B C D are four points on circle.
P is intersection of line AB circumcircle triangle P1P2P3.
```

一个几何命题由三部分组成：

the name, the hypothesis and the conclusion.

具体如下：

Example [name]. $s_1, \cdots, s_r$ CONC.

1° s_i 是一系列几何语句, 他们是几何命题的假设部分.

2° CONC 是几何命题的结论. 可以是以下语句：

```
Show that points P1, P2, and P3 are collinear.
Show that points P1, P2, P3, and P4 are cocircle.
Show that point P1 is the midpoint of P2P3.
Show that [line] P1P2 is parallel to [line] P3P4.
Show that [line] P1P2 is perpendicular to [line] P3P4.
Show that [segment] P1P2 is congruent to [segment] P3P4.
Show that angle P1P2P3 is congruent to angle P4P5P6.
Show that P1 and P2 are inversion with circle P3P4.
Show that P1, P2, P3, P4 form a harmonic sequence.
Show that obj1 is tangent to obj2,
  where obj1 and obj2 could be a line or a circle.
P1 = P2
P1P2=P3P4
P1P2*P3P4 = P5P6*P7P8
exp1 = exp2, where exp1 and exp2 are two algebraic expressions.
```

索　引

Y

Z

其　他